INTELIGENCIA ARTIFICIAL

PRINCIPIOS Y APLICACIONES

Hugo A. Banda Gamboa, MSc, PhD.

Quito, Abril 2014

Información del Autor



Hugo A. Banda Gamboa

Nació en Quito el 7 de Julio de 1950. Sus estudios primarios los realizó en el Centro Experimental Eloy Alfaro. La educación secundaria fue en el Colegio Experimental Central Técnico, en el que se graduó de Bachiller Técnico en Radiotecnia. Ingresó a la Escuela Politécnica Nacional, en donde obtuvo el título de Ingeniero en Electrónica y Telecomunicaciones (1975).

Los estudios de cuarto nivel los realizó en el Reino Unido de la Gran Bretaña. La Universidad de Bradford, Inglaterra, le confirió el Grado de Master of Science (**MSc**) en Power Electronics (1979); y, La Universidad de Dundee, Escocia, le otorgó el Grado de Philosophy Doctor (**PhD**) en Computer Science (Intelligent Systems), en 1990.

Adicionalmente ha realizado estudios superiores en Dirección Universitaria, Pedagogía y Educación Superior en la Universidad Tecnológica Equinoccial (2001), Relaciones Universidad – Industria, Fundación CEDDET, España, 2009.

ACTIVIDADES ACADÉMICAS Y PROFESIONALES

- Presidente de la Corporación Ecuatoriana para la Innovación Científica y Tecnológica (**CORDICYT**), desde Julio 2007.
- Presidente Ejecutivo de la Compañía **SofDecision Software y Sistemas Ltda**, Septiembre 2007-Mayo 2009.
- Presidente de la Sociedad Ecuatoriana de Inteligencia Computacional (**SIEC**), desde Febrero 2007
- Director de Investigación Científica, **SENACYT/FUNDACYT,** 2005-2006.
- Director de la Unidad de Investigación en Sistemas Inteligentes y de Información Geográfica (**UNISIG**), de la EPN, 1994-1999; 2003-2005.
- Presidente de la **Fundación Pro-Ciencia**, para el desarrollo integral del humanismo y del talento científico, con sede en la EPN, 2003-2005
- Profesor Principal (Ciencias de Computación, Sistemas Inteligentes).
- Profesor de los programas de Maestría en Gerencia Empresarial (**MBA**); Maestría en Gestión de las Tecnologías de la Información y las Comunicaciones (**MGTIC**); Maestría en Gestión Tecnológica; Maestría en Informática Educativa; y, en Ingeniería de Sistemas Informáticos y de Computación.
- Consultor en proyectos de aplicación de la informática y de los sistemas inteligentes a la gestión universitaria, empresarial y a la administración de gobiernos locales.
- Proyectos de investigación relacionados con la Gobernanza Universitaria y su Desarrollo en el Siglo XXI; la Universidad Virtual y Aplicaciones Multimedia en la Educación a Distancia; y, Sistemas Inteligentes para Soporte a la Decisión.

PUBLICACIONES

- Varias publicaciones en revistas nacionales e internaciones en tópicos relacionados con los Sistemas de Control, Informática y Comunicaciones, Sistemas Inteligentes y Educación Superior.

hugo.banda@epn.edu.ec
hugo.banda@cordicyt.org

Prefacio

La presente obra constituye una recopilación y actualización de un conjunto de publicaciones que en los últimos 20 años he realizado en la Escuela Politécnica Nacional acerca de diversos tópicos relacionados con la Inteligencia Artificial: Redes Neuronales y Sistemas Expertos, Fundamentos de las Redes Neuronales, Programación Básica en LISP, Principios de la Inteligencia Artificial, Programación en PROLOG, Tópicos Avanzados de Inteligencia Artificial, entre otras.

Esta es una guía a través de los apasionantes tópicos de inteligencia artificial y sus principales aplicaciones. Ha sido escrita a un nivel introductorio para estudiantes de carreras de ingeniería y no requiere más prerrequisitos que fundamentos matemáticos y lenguajes de programación. La presente edición incluye múltiples citas a fuentes de información, las mismas que constan como pies de página para que el lector interesado las utilice para ir construyendo su conocimiento de acuerdo a sus propios intereses.

El contenido se ha organizado de manera secuencial, iniciando con una visión histórica y pasando por los tópicos clásicos de la inteligencia artificial, avanza hacia la inteligencia computacional, para concluir con los sistemas difusos y los agentes inteligentes. En lo posible se han incluido aplicaciones que el estudiante las puede replicar en laboratorio, utilizando software libre y la poderosa herramienta MATLAB, tan popular en carreras de ingeniería.

Dejo constancia de mi agradecimiento a mi esposa Paty, mis hijos Hugo y Kevin por su paciencia y apoyo constante, a los revisores del texto, así como a la Escuela Politécnica Nacional por el financiamiento para la escritura de esta obra, mediante la concesión del permiso con sueldo por un semestre, como lo establece la normatividad institucional.

Atentamente,

Dr. Hugo A. Banda Gamboa
Profesor Principal
Departamento de Informática y Ciencias de Computación

Quito, Abril de 2014

Contenido

Índice de Figuras

1 CAPITULO I: FUNDAMENTOS DE LA INTELIGENCIA ARTIFICIAL

1.1 Introducción

En esta unidad se presentan elementos del marco referencial de la inteligencia artificial; y, partiendo de una breve visión retrospectiva, un análisis actual y futuro de este campo científico relativamente joven.

La inteligencia artificial (IA) es una disciplina muy amplia que comprende tópicos que van desde la filosofía hasta la ciencia y la tecnología. Esto implica que para sistematizar el estudio, es necesario incorporar un marco referencial que facilite la identificación de bloques de conocimiento que vayan aportando a la comprensión de la complejidad teórica, conceptual y contextual de la inteligencia artificial.

1.2 Elementos del Marco Referencial

Toda disciplina científica está inscrita en un marco referencial que incluye aquello en que un investigador puede apoyarse para desarrollar un proyecto:

- El **Marco Contextual** considera características, dimensiones temporales y espaciales del ámbito en el que se inscribe el problema identificado, utilizando para ello:
  - **El Análisis Retrospectivo (Visión histórica)**
  - **El Análisis Situacional (Visión actual)**
  - **El Análisis Prospectivo (Visión futura)**
- **El Marco Teórico – Conceptual**, que incorpora las corrientes epistemológicas planteadas por uno o varios autores, asociadas al significado de los términos (lenguaje técnico) que son necesarias para fundamentar y sustentar las diferentes fases del conocimiento científico asociado al problema identificado (observación, descripción, explicación y predicción).

1.3 Contexto de la Inteligencia Artificial

El estado actual de la inteligencia artificial no es más que el resultado de una larga e intensa búsqueda emprendida por el ser humano para crear seres artificiales capaces de realizar tareas inteligentes, ya sea a su imagen y semejanza o simulando el comportamiento de otras formas de vida. Es por esto que la inteligencia artificial está rodeada de misticismo, teorías, creencias, promesas y esperanzas de que con el apoyo de la ciencia y la tecnología, la humanidad podrá lograr un nuevo génesis de clones y autómatas artificiales.

1.3.1 Historia de la Inteligencia Artificial

La historia de la inteligencia artificial es una colección de mitos, fantasías, mecanismos, teorías y ensayos. Las ideas de máquinas inteligentes y las raíces intelectuales de la inteligencia artificial, aparecen en la antigua cultura griega y otras del medio oriente:

- En la mitología griega, **Hefestos,** dios del fuego y la forja, creó diversas criaturas: ***Talos***, el gigante de bronce que Zeus dio a Europa para que fuese el guardián de Creta; las ***Kourai Khryseai*** (doncellas doradas), dos autómatas de oro con la apariencia de jóvenes mujeres vivas, que poseían inteligencia, fuerza y el don del habla. Ellas atendían a Hefestos en su palacio del Olimpo.
- En la mitología judía y en el folclore medieval, un ***golem*** es un ser animado fabricado a partir de materia inanimada. Actualmente, el concepto de golem, autómatas y simulacros similares es popular en los juegos para computadora. A través del tiempo ha dado nacimiento a una gran cantidad de variaciones.
- La historia dice que la fabricación de máquinas que imitan al ser humano o a otros seres vivos es incluso anterior a la era cristiana. De hecho, hace unos 4000 años, en el Antiguo Egipto existían estatuas de dioses o reyes que despedían fuego desde sus ojos, tal como la estatua de Osiris. Otros ingenios estaban dotados de brazos mecánicos que eran operados por los sacerdotes del templo.
- El autómata que más llamó la atención de los antiguos fue la estatua de **Memon de Etiopía**, que era capaz de emitir sonidos cuando los rayos del sol la iluminaba, causando el lógico temor y el respeto entre la población. No había nada de mágico en el autómata: al amanecer, el cambio de temperatura provocaba la evaporación del agua almacenada en su interior, y era ese vapor el que al escapar por fisuras muy bien calculadas de la estatua producía un sonido semejante al habla.
- Desde la perspectiva intelectual, en el Siglo IV antes de Cristo, **Aristóteles** fundó la **lógica silogística**. Sus trabajos principales sobre la materia, que tradicionalmente se agrupan bajo el nombre ***Órganon*** (herramienta), constituyen la primera investigación sistemática acerca de los principios del razonamiento válido o correcto. Sus propuestas han ejercido influencia desde hace más de dos milenios, hasta la época actual.
- También los griegos fueron grandes constructores de autómatas. Empleaban a menudo la energía hidráulica para dotar de movimiento a sus creaciones. **Herón de Alejandría** (contemporáneo de Cristo) escribió un libro, llamado **Autómata**, en el que describen mecanismos construidos con fines de entretenimiento capaces realizar variados movimientos. Entre los mencionados se destacan los que imitaban aves que gorjean, vuelan y beben; estatuas capaces de servir una copa de vino o puertas automáticas. El motor de estas máquinas antiguas era el

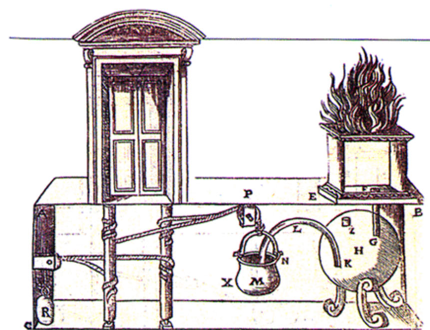


Figura 1. Mecanismo de apertura de puerta (Ilustración del libro Autómata)

movimiento del agua, la fuerza de gravedad o bien algún sistema de palancas. De esa época data también un teatro de marionetas mecánicas capaz de representar escenas de la Guerra de Troya. [1]

- Entre el Siglo XII y el Siglo XIII, **Al-Jazari**, importante erudito, artista, astrónomo, inventor e ingeniero mecánico proveniente de Al-Jazira (Mesopotamia) que floreció durante la Edad de Oro del Islam en la Edad Media, sirvió de ingeniero principal en el Palacio Artuklu. En 1206 publicó, **El Libro del Conocimiento de los Ingeniosos Mecanismos**. En él se recopilaban muchos mecanismos ingeniosos de los siglos anteriores, a la vez que se exponían algunos creados por el autor: un reloj de agua accionado por contrapesos y un autómata de forma humanoide que era capaz de servir bebidas. [2]

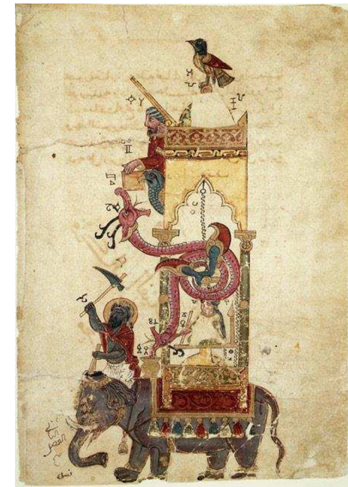


Figura 2.Reloj Elefante de Al-Jazari

- Las primitivas máquinas parlantes, fueron consideradas como trabajos heréticos de magos que intentaban desafiar a Dios. En el Siglo XIII, el Filósofo **Albertus Magnus**, se dice que creó una cabeza que era capaz de hablar, que luego fue destruida por su discípulo Santo Tomás de Aquino, que la consideró como una abominación. El monje científico inglés, **Roger Bacon** también parece que logró producir una versión. [3]



Figura 3. Cabeza parlante

- En 1495, **Leonardo Da Vinci**, construyó un robot capaz de pararse, sentarse, caminar, abrir la boca, mover la cabeza y levantar los brazos. Para esto utilizó sus estudios realizados en anatomía y kinestésica. Su creación fue una extensión de la hipótesis que tenía de que el cuerpo humano era una máquina en estructura, cuyos complicados movimientos podían ser imitados mediante partes mecánicas de ingeniería tales como piñones, palancas y poleas (piezas de relojería). También hay relatos que en 1515 construyó para el Rey de Francia, un león mecánico que caminaba, movía la cola, abría sus fauces y sacudía su cabeza. Estos y varios otros autómatas diseñados por Leonardo Da Vinci, están documentados en el libro: ***I Robot di Leonardo Da Vinci***, escrito por **Mario Taddei.** [4]
- Durante la primera mitad del siglo XVI, el ingeniero italiano **Giovanni Torriani**, quien fuera Matemático Mayor de la Corte de Felipe II de España, construyó un autómata llamado **El Hombre de Palo**.

---

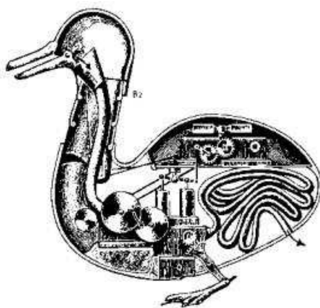
1 http://es.wikipedia.org/wiki/Herón_de_Alejandría
2 http://muslimheritage.com/topics/default.cfm?ArticleID=851
3 http://www.haskins.yale.edu/featured/heads/simulacra.html
4 http://www.leonardo.it/leonardo/press.htm

Según la leyenda, tal era la complejidad del mecanismo creado por Torriani, que su autómata era capaz de cruzar una de las calles de Toledo (a la que posteriormente dio nombre) y pedir limosna para la construcción de un hospital. [5]



Figura 4. Placa en Toledo

- Durante el Siglo XVII, el filósofo **René Descartes**, propone que el cuerpo de los animales se puede describir como máquinas complejas. Otros pensadores ofrecen variaciones y elaboraciones de las propuestas Cartesianas.
- En 1642, **Blaise Pascal** crea la primera máquina calculadora digital mecánica, diseñada para adición y sustracción.
- En 1651, el filósofo inglés **Thomas Hobbes**, publica ***The levathian***, donde propone una teoría del pensamiento mecánico y combinatorio.
- En 1671, el matemático y filósofo alemán **Gottfried Wilhelm von Leibniz**, diseña la máquina calculadora que la denomina ***Step Reckoner***, mejorando la máquina de **Pascal** ya que incluye las operaciones de multiplicación. A pesar que su calculadora representaba los números en forma decimal, **Leibniz** fue uno de los pioneros en reconocer la importancia del sistema binario en las máquinas de cálculo.


Figura 5. Pato con aparato digestivo

- A fines del Siglo XVII e inicios del Siglo XVIII, **Jacques de Vaucanson** construía una serie de autómatas que actuaban como flautistas y tamborileros, que resultaron un gran éxito. Su último y más famoso mecanismo fue el llamado **Pato con Aparato Digestivo**, un dispositivo dotado de más de cuatrocientas piezas móviles capaz de batir las alas, comer, realizar la digestión y defecar tal como lo hace un pato vivo. [6]
- Durante el siglo XVIII el relojero suizo **Pierre Jaquet-Droz** creó obras que causaron gran impresión en los lugares donde fueron exhibidas. Tres de ellas: **La Pianista**, **El Dibujante** y **El Escritor**, pueden verse en el Museo de Arte e Historia de Neuchâtel, en Suiza. Las tres tienen más de 2000 piezas móviles, y **El Escritor** supera las 6000. [7]


Figura 6. Pianista, dibujante y escritor

[5] http://goo.gl/15SOI
[6] http://www.swarthmore.edu/Humanities/pschmid1/essays/pynchon/vaucanson.html
[7] http://en.wikipedia.org/wiki/Pierre_Jaquet-Droz

- En 1818, **Mary Shelley** publica el libro sobre el monstruo ***Frankenstein***.
- Los británicos **Charles Babbage** y **Ada Byron** (**Lady Lovelace**) a mediados de la década de los años 1830, diseñan una máquina calculadora programable mecánica (**Analytical Engine**). Si bien no llegaron a completar su diseño, un modelo operativo basado en sus apuntes no publicados, descubiertos en 1937, fue presentado en 2002. [8]
- El matemático inglés **George Boole**, desarrolla el álgebra binaria y publica en 1853 su obra: ***An Investigation of The Laws of Thought***. [9]
- En 1879 el matemático y logístico alemán **Gottlob Frege**, desarrolla la **lógica proposicional** moderna y publica la obra ***Grundgesetze der Arithmetik*** [10], que posteriormente fue clarificada y expandida por el filósofo Bertrand Russell y los matemáticos **Alfred Tarski**, **Kurt Gödel** y otros.
- Entre 1910 y 1913, **Alfred North Whitehead** y **Bertrand Russell**, publican en tres volúmenes ***Principia Mathematica*** [11], en donde se establecen las bases de la lógica formal y de la moderna lógica matemática.
- En 1920 la palabra robot se adopta para designar a los autómatas. Fue el escritor checo de **Karel Capek**, quien la menciona en su obra de teatro "R.U.R." (*Robots Universales de Rossum*). En checo, *robota* significa *trabajo forzado*.
- En 1943, **Warren McCulloch** y **Walter Pitts** publican *A Logical Calculus of the Ideas Immanent in Nervous Activity*, estableciendo los fundamentos de las redes neuronales artificiales.[12]
- En el artículo ***Behavior, Purpose and Teleology*** publicado en 1943 por **Arturo Rosenblueth**, **Norbert Wiener** y **Julian Bigelow** se acuña el término ***cibernética***. Posteriormente, en 1948, **Norbert Wiener** publica su popular libro titulado ***Cybernetics or Control and Communication in the Animal and the Machine*** [13].
- En 1943, el matemático y logístico polaco **Emil Post** prueba que los Sistemas de Producción son un mecanismo de computación general (Sistema Formal, llamado **Máquina de Post**). Posteriormente, los sistemas de producción facilitan el desarrollo de sistemas expertos basados en reglas.
- En 1945, el matemático **George Pólya** publica su libro ***How to Solve It*** [14], relacionado con el **pensamiento heurístico**, que alcanzó acogida mundial, vendió más de un millón de copias, ha estado en continua impresión desde su primera publicación e influenció en muchos científicos de la inteligencia artificial.
- En Junio de 1945, se publicó el *First Draft of a Report on the EDVAC,* preparado por **John von Neumann**, incluyendo ideas de **J. Presper Eckert** y **John Mauchly**. El informe

[8] http://www.computerhistory.org/babbage/
[9] http://www.gutenberg.org/ebooks/15114
[10] http://www.iep.utm.edu/prop-log/
[11] http://plato.stanford.edu/entries/principia-mathematica/
[12] Warren McCulloch and Walter Pitts, *A Logical Calculus of Ideas Immanent in Nervous Activity*, 1943, Bulletin of Mathematical Biophysics 5:115-133.
[13] http://mitpress.mit.edu/books/cybernetics-or-control-and-communication-animal-and-machine
[14] http://serl.iiit.ac.in/cs6600/book.pdf

contenía una propuesta muy detallada para el diseño de una computadora que desde entonces se conoce como EDVAC (*Electronic Discrete Variable Automatic Computer*)[15]. La arquitectura de computadoras que esbozó adquirió el nombre de **Arquitectura von Neumann**.

- En Julio de 1945, en la revista mensual *The Atlantic*, **Vannevar Bush** publica el artículo ***As We May Think***[16], en el que presenta una visión de cómo los computadores asistirán a los humanos en el futuro.
- En 1950, **Alan Turing** publica ***Computing Machinery and Intelligence*** en este artículo propone una forma de comprobar el comportamiento inteligente (***The Turing Test***). [17]
- En 1950, **Claude Shannon** publica ***Programming a Computer for Playing Chess*** en el que detalla el juego de ajedrez como un proceso de búsqueda. [18]
- **Isaac Asimov** en su libro ***I, Robot***, publicado en 1950, establece las **tres leyes de la robótica**. [19]

1.3.2 La Conferencia de Dartmouth

El inicio de la era actual de la inteligencia artificial, se sitúa en el año de 1956, a partir de la histórica **Conferencia de Dartmouth**[20], en la que participaron, entre otros, **John McCarthy**, **Marvin Minsky**, **Claude Shannon**, **Allen Newell** y **Herbert Simon**. En esta conferencia que tuvo un mes de duración, se acuñó el término inteligencia artificial y se ensayó la primera definición:

> ***Hacer que una máquina se comporte como lo haría un ser humano, de tal manera que se la podría llamar inteligente.***

1.3.3 La Inteligencia Artificial en la Segunda Mitad del Siglo XX

A fines de 1956, **Allen Newell**, **J.C. Shaw** y **Herbert Simon** (Carnegie Institute of Technology, ahora Carnegie Mellon University) realizaron la demostración de la ejecución del primer programa de inteligencia artificial, ***The Logic Theorist***. Este mismo grupo de investigadores, en 1957 presentaron la aplicación inteligente: ***The General Problem Solver***.

En 1958, **John McCarthy** inventa **LISP**, el primer lenguaje para inteligencia artificial. **Herb Gelernter** y **Nathan Rochester** (IBM) describen un demostrador de teorema de geometría que explota un modelo semántico del dominio en forma de diagramas de casos típicos. La Conferencia de Teddington en la mecanización de los procesos de pensamiento se celebró en el Reino Unido y entre las ponencias presentadas estuvieron: *Programs with Common Sense*, de **John McCarthy**, *Pandemonium* de **Oliver Selfridge**; y, *Some Methods of Heuristic Programming and Artificial Intelligence* de **Marvin Minsky**.

---

15 http://es.wikipedia.org/wiki/EDVAC
16 http://www.theatlantic.com/magazine/archive/1945/07/as-we-may-think/3881/
17 http://blog.santafe.edu/wp-content/uploads/2009/05/turing1950.pdf
18 http://vision.unipv.it/IA1/ProgrammingaComputerforPlayingChess.pdf
19 http://enciclopedia.us.es/index.php/Tres_leyes_de_la_robótica
20 http://en.wikipedia.org/wiki/Dartmouth_Conferences

En los siguientes años, hasta finales del Siglo XX, se desarrollan una serie de propuestas y aplicaciones de inteligencia artificial, entre los que se pueden destacar:

- **SAINT**, el primer programa de integración simbólica, escrito en LISP, para la solución de problemas de cálculo de nivel colegial (**James Slagle**, PhD dissertation, MIT, 1961).
- En el MIT, **Danny Bobrow**, en su disertación, demuestra que equipos pueden entender lenguaje natural lo suficientemente bien como para resolver correctamente problemas verbales de álgebra. **Bert Raphael**, en su disertación, demuestra con el programa **SIR** el poder de la representación lógica de conocimientos para sistemas basados en preguntas y respuestas (1964).
- En 1965, **John Alan Robinson**, inventó el **Método de Resolución** que permitió a programas trabajar de forma eficaz usando la lógica formal como lenguaje de representación. **Joseph Weizenbaum,** en el MIT construyó a ***ELIZA***, un programa interactivo que dialogaba en inglés sobre cualquier tema. Una versión que simulaba el diálogo con un psicoterapeuta, fue un juguete muy popular en los centros de IA de la red ARPA.
- En 1966, **Ross Quillian** (Carnegie INST de tecnología; ahora CMU) en su disertación demostró la utilidad de las **redes semánticas** en aplicaciones de IA. En este mismo año, el *Automatic Language Processing Advisory Committee* (ALPAC) publica un informe negativo sobre traducción automática de máquina que detendrá por muchos años (hasta fines de la década de los años 80) la investigación en Procesamiento de Lenguaje Natural (NLP) [21].
- En 1967, **Edward Feigenbaum**, **Joshua Lederberg**, **Bruce Buchanan**, **Georgia Sutherland** publican en Stanford el programa **Dendral**, capaz de interpretar la espectrometría de masas en compuestos químicos orgánicos. Este fue el primer programa **basado en conocimiento** para razonamiento científico. **Joel Moses** en su trabajo de doctorado en el MIT, con el programa **Macsyma**, demostró el poder de razonamiento simbólico para los problemas de integración. Fue el primer programa exitoso, basado en conocimientos, para matemáticas. **Richard Greenblatt** en el MIT construyó un programa basado en conocimiento para jugar ajedrez, **MacHack**.
- En 1968, **Marvin Minsky** y **Seymour Papert** publican el libro ***Perceptrons***, que se convirtió en el trabajo fundacional en el análisis de redes neuronales artificiales. Su crítica de la investigación poco rigurosa en el campo, ha sido considerada como responsable de la desaparición virtual de la investigación académica en redes neuronales artificiales durante los años 70 (***AI Winter***).
- En 1969, **Carl Hewitt** en el MIT, publica el lenguaje de programación **Planner**. Este lenguaje fue una especie de híbrido entre los paradigmas procedimentales y lógicos porque combinaba facilidad de programación con razonamiento lógico. En primer

---

[21] ALPAC (1966): *Language and machines: computers in translation and linguistics*. A report by the Automatic Language Processing Advisory Committee, Division of Behavioral Sciences, National Academy of Sciences, National Research Council. Washington, D.C.: National Academy of Sciences - National Research Council.

lugar, se implementaron subconjuntos como **Micro-Planner** y **Pico-Planner**, y luego el lenguaje completo fue implementado. Derivaciones, como QA4, **Conniver**, **QLISP** y **Ether** fueron elementos importantes en la investigación de Inteligencia Artificial en la década de 1970, que influyó en las herramientas comerciales de desarrollo **KEE** y **ART**. El centro de Inteligencia Artificial del ***Stanford Research Institute*** (**SRI**) presenta a ***Shakey,*** el primer robot móvil capaz de razonar sobre sus acciones. **Shakey** dejó un importante legado e influencia en la inteligencia artificial y la robótica actual[22].

- En 1970, **Jaime Carbonell** desarrolló **SCHOLAR**, un programa interactivo para la instrucción asistida por computador, basado en **redes semánticas** para la representación del conocimiento. **Bill Woods** describió las **Redes de Transición aumentada** (**ATN**) como representación para la comprensión del lenguaje natural. En el **MIT**, el programa **ARCH** desarrollado por **Patrick Winston** en su PhD, aprendía conceptos a partir de ejemplos, en el mundo de los bloques de niños.
- En 1971, la tesis de PhD de **Terry Winograd** (MIT) demostró la habilidad de los computadores para comprender frases en inglés en el mundo restringido de los bloques de niños. Su programa **SHRDLU**, acoplado a un brazo robótico le permitía ejecutar instrucciones escritas en inglés. [23]
- En 1972, **PROLOG**, el lenguaje de inteligencia artificial basado en lógica formal, fue desarrollado por **Alain Colmerauer**.
- En 1973, el grupo *Assembly Robotics* de la Universidad de Edimburgo construye **Freddy**, el famoso robot escocés, capaz de usar la visión para localizar y ensamblar modelos.
- En 1974, la tesis de doctorado de **Ted Shortliffe** en Stanford, desarrolló **MYCIN**, el primer sistema experto, demostrando el poder de los sistemas basados en reglas para la representación del conocimiento y la inferencia en el dominio de diagnóstico médico y terapia[24]. También en Stanford, **Earl Sacerdoti** desarrolló uno de los primeros programas de planificación, el ***Abstraction-Based Stanford Research Institute Problem Solver*** (**ABSTRIPS)** y las técnicas de planificación jerárquica. [25]
- En 1975, **Marvin Minsky** publicó su artículo ampliamente leído e influyente acerca de los ***frames*** como una representación del conocimiento, en el que se reúnen muchas ideas acerca de esquemas y vínculos semánticos.[26] En Xerox PARC, **Alan Kay** y **Adele Goldberg** desarrollaron el lenguaje **Smalltalk**, estableciendo el poder de la programación orientada a objetos y de interfaces basadas en iconos. [27]
- En 1979, la tesis de **Bill VanMelle** en Stanford demuestra la generalidad de la representación de conocimiento y estilo de razonamiento de **MYCIN**, en su programa **EMYCIN**, que sirvió de modelo para muchos sistemas comerciales de *shells* expertos.

[22] http://www.sri.com/about/timeline/shakey.html
[23] http://hci.stanford.edu/~winograd/shrdlu/
[24] Shortliffe, E.H., *Computer Based Medical Consultations: MYCIN*, American Elsevier, 1976.
[25] "Planning in a Hierarchy of Abstraction Spaces," *Artificial Intelligence,* Vol. 5, No. 2, pp. 115-135 (1974).
[26] Minsky Marvin. *FRAMES: A Framework for Representing Knowledge*. In, *The Psychology of Computer Vision*, P. Winston (Ed.), McGraw-Hill, 1975.
[27] http://www.clubsmalltalk.org/web/

Jack Myers y **Harry Pople** en la Universidad de Pittsburgh desarrollaron el **INTERNISTA**, un programa de diagnóstico médico basado en el conocimiento clínico del **Dr. Myers**. **Cordell Green, David Barstow, Elaine Kant** y otros en Stanford demostraron el sistema **CHI** para **programación automática**.

- A finales de la década de 1970, se inicia un ambicioso proyecto propuesto por Japón: ***The Fifth Generation Computer Systems***[28]. Su objetivo fue el desarrollo de una nueva clase de computadoras que utilizarían técnicas y tecnologías de Inteligencia Artificial tanto en hardware como en software, incluyendo PROLOG como lenguaje de máquina. Estos sistemas serían capaces de resolver problemas complejos, como la traducción automática de una lengua natural a otra. El proyecto duró once años, pero no obtuvo los resultados esperados.
- Como respuesta al proyecto japonés, en otros países se crearon proyectos paralelos. En Estados Unidos, la ***Microelectronics and Computer Technology Corporation*** y la ***Strategic Computing Initiative***; en el Reino Unido fue el Proyecto **ALVEY** y en el resto de Europa su reacción fue conocida como **ESPRIT** (***European Strategic Programme for Research in Information Technology***[29]). Éstos, al igual que el programa japonés, finalizaron una década más tarde, sin mayor éxito.
- En 1980, el ***Stanford Cart***, construido por **Hans Moravec** [30], se convierte en el primer vehículo autónomo controlado por computador. Atravesó una sala llena de sillas con éxito y circunvaló el laboratorio de IA de Stanford. En esta década, se desarrollaron y comercializaron **Máquinas LISP**, así como los primeros *shells* para desarrollo de sistemas expertos y aplicaciones comerciales. **Lee Erman, Rick Hayes-Roth, Victor Lesser** y **Raj Reddy** publicaron la primera descripción del **modelo de pizarra**, como marco para **HEARSAY-II**, el sistema de comprensión del lenguaje hablado. [31]
- En 1981, **Danny Hillis** diseña la ***Connection Machine***, una arquitectura masivamente paralela que trae nuevo poder a la IA y a la computación en general. (Más tarde funda *Thinking Machines, Inc*. [32])
- En 1983, **John Laird** y **Paul Rosenbloom**, trabajando con **Allen Newell**, completaron las disertaciones acerca de **SOAR**, una arquitectura cognitiva general, para el desarrollo de sistemas que exhiben un comportamiento inteligente[33]. **James Allen** inventa el **cálculo de intervalo**, la primera formalización ampliamente utilizada para eventos temporales.

---

[28] Moto-oka, Tohru. *Overview to the Fifth Generation Computer System project*. Proceedings of the 10th annual international symposium on Computer architecture. ACM New York, USA, 1983, Pages 417-422

[29] https://aclweb.org/anthology/H/H91/H91-1007.pdf

[30] Moravec, Hans Peter, *Obstacle avoidance and navigation in the real world by a seeing robot rover*, PhD in Computer Science, 1980.

[31] Lee Erman, Rick Hayes-Roth, Victor Lesser and Raj Reddy. *The Hearsay-II Speech-Understanding System: Integrating Knowledge to Resolve Uncertainty*. ACM, Computing Surveys, Vol 12, No. 2, June 1980, pp 213-253

[32] http://www.youtube.com/watch?v=IjmostrFetg

[33] http://sitemaker.umich.edu/soar/home

- A mediados de los 80 aparecieron una serie de aplicaciones basadas en redes neuronales artificiales, entrenadas por el algoritmo de ***Backpropagation*** (descrito por primera vez por **Paul J. Werbos** en 1974[34]).
- En 1986, **Marvin Minsky** publica ***The Society of Mind***, una descripción teórica de la mente, como una colección de agentes cooperantes[35].
- En 1989, **Dean Pomerleau,** en Carnegie Mellon University, crea **ALVINN** (un vehículo autónomo terrestre, que utiliza una red neuronal). Posteriormente se convirtió en un sistema que condujo un coche de costa a costa (unos 2850 Km.), bajo el control de una computadora.

Durante los años 90, se dieron avances en todas las áreas de AI, con importantes manifestaciones en aprendizaje automático, tutoría inteligente, razonamiento basado en casos, planificación multi-agente, programación, razonamiento incierto, *data mining*, comprensión del lenguaje natural y traducción, visión, realidad virtual, juegos y otros temas:

- El proyecto **COG** de **Rod Brooks** en el MIT, con varios colaboradores, hace progresos significativos en la construcción de un **robot humanoide**.
- **TD-Gammon**, un programa de *backgammon* escrito por **Gerry Tesauro**, demuestra que el **aprendizaje por refuerzo** es lo suficientemente poderoso como para crear un programa de juegos de nivel de campeonato capaz de competir favorablemente con jugadores de clase mundial.
- El **Demostrador de Teoremas EQP** (Laboratorio Nacional de Argonne, Illinois, USA) demuestra la conjetura de Robbins en matemáticas (1996).
- El programa de ajedrez ***Deep Blue*** vence al entonces campeón mundial de ajedrez, **Garry Kasparov**, en un partido y su revancha, vistos mundialmente[36] (1997).
- En 1997, Nagoya, Japón, se realiza el primer torneo oficial de Fútbol **Robo-Cup**, con 40 equipos de robots que interactúan en juegos de mesa, ante la presencia de más de 5000 espectadores. [37]
- Los navegadores Web y otros programas para la extracción de información basados en IA se convierten en algo esencial para el uso generalizado de la *World Wide Web*.
- En el laboratorio de IA del MIT, se realiza la demostración de una **sala inteligente** y **agentes emocionales**. Se inician los trabajos del denominado ***Project Oxygen***, que conecta equipos móviles y estacionarios, en una red adaptable. [38]

---

[34] Werbos, P J. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD thesis, Harvard University, 1974.
[35] Minsky, Marvin. *The Society of Mind*. New York: Simon and Schuster, 1986.
[36] http://www.research.ibm.com/deepblue/home/html/b.html
[37] http://www.sonycsl.co.jp/person/kitano/RoboCup/RoboCup-old.html
[38] http://oxygen.lcs.mit.edu/Overview.html

1.3.4 La Inteligencia Artificial a Inicios del Siglo XXI

En los años 2000, aparecen versiones comerciales de mascotas robóticas interactivas, (también conocidas como *juguetes inteligentes*) actualizando la visión de los fabricantes de juguetes novedosos del Siglo XVIII. **Cynthia Breazeal** en el MIT publica su tesis doctoral en ***Sociable Machines***, describiendo **KISMET**, un robot con un rostro que expresa emociones[39]. **Stanley**, el vehículo autónomo de la Universidad de Stanford, gana la carrera *Grand Challenge* organizada por DARPA (Octubre de 2005)[40]. El robot **Nomad** explora las regiones remotas de la Antártida buscando muestras de meteorito. [41]

Esta breve visión histórica de la IA, no estaría completa sin considerar el desarrollo actual de la robótica japonesa, china y sur coreana.

1.3.4.1 Robótica Japonesa

Japón tiene una larga historia de interés en la robótica. Entre los precursores del robot más antiguo en Japón están los ***Karakuri Ningyo***, o muñecos mecánicos. Los *Karakuri Ningyo* se cree que se originaron en China, durante el período Edo (1603-1867). Una de las características y avances de la robótica japonesa en relación a los de otros países es la movilidad de los robots desarrollados. Los robots humanoides japoneses incluyen habilidades tales como parpadear, sonreír o expresar emociones tales como la ira y la sorpresa. Los robots diseñados para jugar con los niños generalmente se parecen a los animales y en función de ello, hacen diferentes sonidos, se mueven, caminan y juegan.

Los científicos japoneses, han desarrollado diversos tipos de robots:[42]

- Robots humanoides para entretenimiento
  - ASIMO , fabricado por Honda
  - QRIO, de Sony
  - HOAP de la serie Robot humanoides de plataforma de arquitectura abierta, fabricado por Fujitsu
  - Toyota Partner Robot , fabricado por Toyota
  - EMIEW, de Hitachi
- Androides (Robots diseñados para parecerse a los seres humanos)
  - Actroid, un robot femenino muy realista demostrado en la Expo 2005 en Japón
  - Hanako, un robot humanoide diseñado para entrenamiento de un dentista
  - HRP-4C , un robot humanoide con una cabeza muy realista y la figura media de una joven mujer japonesa
- Animales robóticos (cuatro patas)
  - AIBO es un perro robot comercial fabricado por Sony Electronics
- Robots sociales

---

[39] http://www.ai.mit.edu/projects/sociable/baby-bits.html
[40] http://cs.stanford.edu/group/roadrunner/stanley.html
[41] http://www.frc.ri.cmu.edu/projects/meteorobot2000/
[42] http://en.wikipedia.org/wiki/Japanese_robotics

    - PaPeRo
    - Paro, una foca robot destinada a usos terapéuticos
    - Wakamaru
- Robots guardianes
    - Guardrobo D1 es fabricado por Sohgo Security Services
    - Banryu , fabricado por Sanyo y Tmsuk
- Robots domésticos
    - SmartPal V, fabricado por Yaskawa Electric Corporation
    - Twendy-ONE , desarrollado por la Universidad de Waseda
    - TPR-Robina , fabricado por Toyota
- Robot de movilidad
    - WL-16RIII, desarrollado por la Universidad de Waseda y Tmsuk
    - i pies, desarrollado por Toyota
    - i-REAL , desarrollado por Toyota
    - Murata Boy, desarrollada por Murata Manufacturing Co. Ltd.
- Robots de rescate
    - T-53 Enryu , fabricado por Tmsuk
- Robótica Humanoide Industrial
    - HRP-3 PROMET Mk-II, fabricado por Industrias Kawada, diseñado por Yutaka Izubuchi
    - HRP-4
- Robótica Industrial
    - El dominio y expansión de la robótica industrial japonesa alrededor del mundo, se han visto favorecidos por los recursos financieros disponibles a largo plazo y el fuerte mercado interno que gozan las empresas japonesas. Sólo pocas empresas no japonesas han logrado sobrevivir en el mercado mundial, incluyendo *Adept Technology*, *Stäubli Unimation*, la empresa suiza-sueca ABB (*Asea Brown Boveri*), el fabricante austriaco *igm Robotersysteme AG* y la empresa alemana *KUKA Robotics*. Esto incluye los utilizados en las plantas de producción de automotores, conocidos como robots de línea de montaje.

1.3.4.2 Robótica China

La investigación china en el campo de la robótica se inicia en los años 70 y avanzó muy lentamente en los primeros 10 años. A partir de 1985, se acelera el desarrollo y para inicio del Siglo XXI, los expertos chinos en robótica establecen dos líneas de aplicación, de acuerdo con el entorno de actuación: Robots industriales (manipuladores y multiangulares) y Robots Especiales (de servicio, subacuáticos, militares, agrícolas y de micro-operación).

A partir del año 2000, sus productos robóticos han logrado captar un interesante mercado mundial (Con un valor de 1.000 millones de Yuans, en el 2010), basados en bajo costo, buena tecnología, alto desempeño y competitividad. [43, 44]

1.3.4.3 Robótica Sur Coreana

Corea del Sur es una de las sociedades que usa la más alta tecnología y conexiones de mayor ancho de banda del mundo. Los ciudadanos disfrutan de tecnología móvil avanzada mucho antes de que ésta llegue a los mercados occidentales. El Gobierno también es conocido por su compromiso con la tecnología del futuro. Un reciente informe del Gobierno prevé que robots realizarán rutinariamente cirugía para el 2018. El Ministerio de información y la comunicación también ha predicho que cada hogar de Corea del Sur tendrá un robot entre el 2015 y 2020. En parte, esto es una respuesta al envejecimiento de la sociedad del país y también el reconocimiento que el ritmo de desarrollo en robótica se está acelerando. [45]

1.3.5 Situación Actual

Transcurrido más de medio siglo desde la Conferencia de *Dartmouth* en 1956, fecha mayoritariamente reconocida como el inicio de la inteligencia artificial, ésta se ha constituido en una disciplina sumamente amplia. Contiene muchos y diversos sub campos y tiene fuertes vínculos con áreas como la ciencia cognitiva y la filosofía de la mente.

La investigación de inteligencia artificial abarca una amplia gama de temas. Incluye la mejora de computadoras en tareas que son ampliamente percibidas como inteligentes, como probar teoremas matemáticos. También investiga procesos que recientemente se han reconocido tienen dificultad importante, como el reconocimiento de objetos en imágenes. Investiga problemas que no se habían considerado antes de la invención de las computadoras, tales como procesar grandes cantidades de datos y encontrar información en la *World Wide Web*.

El campo de la Inteligencia Artificial no sólo se preocupa por replicar la inteligencia humana. Sistemas de inteligencia artificial, tales como programas de juego de ajedrez o que incluyen procesos de decisión, no se comportan necesariamente como lo harían los humanos, sin embargo han demostrado habilidad para resolver problemas complejos. Hay todo un campo independiente, la ciencia cognitiva, que se dedica a comprender cómo los seres humanos (y otros animales) piensan. Hay muchos vínculos fértiles entre los dos campos. En muchos casos una mejor comprensión de la cognición humana puede llevar a avances en Inteligencia Artificial. Al mismo tiempo encontrar una forma para que un computador pueda realizar una tarea, puede arrojar luz sobre la forma en la que los seres humanos podrían pensar.

---

[43] Hongliang Cui. *Robotics in China*. Power Point presentation, http://www.authorstream.com
[44] http://robotics.youngester.com/2010/07/growth-in-robotics-in-china-china-is.html
[45] http://www.weirdasianews.com/2008/04/29/south-korea-to-build-worlds-first-robot-themed-parks/

Por otra parte, no es sorprendente que la Inteligencia artificial, sea la inspiración para mucha ficción especulativa. A menudo se trata de historias en las que computadoras o robots se comportan como versiones especialmente inteligentes y físicamente más fuertes que los seres humanos. Hay una serie de problemas prácticos y filosóficos detrás de estas ideas. Todavía no es totalmente claro si la naturaleza de la investigación de la Inteligencia Artificial podrá producir este tipo de robot inteligente y malévolo.

Los campos de la Inteligencia Artificial y las Ciencias Cognitivas son importantes componentes de las Ciencias de la Computación y la Psicología. El desafío de la Inteligencia Artificial incluye desarrollar en las máquinas la capacidad de percibir, aprender, almacenar información, razonar acerca de lo que conocen, comunicarse utilizando lenguaje humano e interactuar con el entorno físico.

Un ejemplo en esta línea es el **Proyecto Watson** desarrollado por la **IBM**. Watson es un sistema informático de inteligencia artificial, capaz de responder a preguntas formuladas en lenguaje natural. Forma parte del proyecto del equipo de investigación **DeepQA**, liderado por el investigador **David Ferrucci**. Lleva su nombre en honor del fundador de IBM, **Thomas J. Watson**. Watson responde a las preguntas gracias a una base de datos almacenada localmente. La información contenida en esa base de datos (IBM DB2), proviene de una multitud de fuentes, incluyendo enciclopedias, diccionarios, tesauros, artículos de noticias, obras literarias, taxonomías, etc. [46]

A lo largo de tres días, en febrero de 2011, para probar sus capacidades reales Watson participó de un programa especial de juegos en el concurso de televisión estadounidense Jeopardy!, derrotando a sus dos oponentes humanos: **Brad Rutter** (el mayor ganador de dinero en toda la historia del programa), y **Ken Jennings** (el poseedor del récord por haber ganado 75 partidas seguidas)[47]. Watson recibió el primer premio de $1.000.000, mientras **Ken Jennings** y **Brad Rutter** recibieron $300.000 y $200.000, respectivamente[48].

Todas estas habilidades todavía están siendo estudiadas por los investigadores y ninguna puede considerarse resuelta. Esto hace que la Inteligencia Artificial y las Ciencias Cognitivas sean importantes campos de estudio.

1.4 Marco Teórico – Conceptual de la Inteligencia Artificial

La inteligencia Artificial es una parte central de uno de los más profundos avances científicos e intelectuales de los últimos 60 años: el estudio de la información y del conocimiento; y, cómo estos pueden ser adquiridos, almacenados, entendidos, manipulados, usados y transmitidos. La investigación ha permitido desarrollar una nueva ciencia para el estudio de los principios por los cuales: conocimiento es adquirido y

[46] http://es.wikipedia.org/wiki/Watson_(inteligencia_artificial)
[47] http://youtu.be/97oecf5S6kA
[48] http://youtu.be/D2H5j_UYtSM

utilizado, metas son generadas y logradas, información es comunicada, colaboración es alcanzada, conceptos formados, y el lenguaje es utilizado. A ésta se la puede llamar la ciencia del conocimiento o la ciencia de la inteligencia. Igualmente, la creciente necesidad de desarrollar aplicaciones capaces de resolver problemas complejos, ejecutando procesos que se pueden llamar inteligentes, ha dado lugar al aparecimiento de la ingeniería de sistemas inteligentes. Acorde con estas consideraciones, en términos simples, se puede decir que:

> *La **Inteligencia Artificial** comprende la investigación científica y tecnológica de los **sistemas inteligentes**.*
>
> *Se denomina **sistema inteligente** a una **entidad** capaz de percibir, razonar, aprender, adaptarse, tomar decisiones y actuar racionalmente para satisfacer sus metas, en un determinado entorno.*
>
> *Las **entidades** pueden ser: máquinas, humanos u otros animales*

Dada su complejidad, el desarrollo de la investigación científica y tecnológica de los sistemas inteligentes, requiere del apoyo de otras áreas del conocimiento, tales como: Filosofía, Psicología, Lingüística, Ciencias de Computación, Biología, Neurociencias, Matemática, Física, Química, Cibernética, Electrónica y Comunicaciones.

Desde la perspectiva filosófica, muchos teóricos de IA creen que ciertos procesos computacionales son necesarios y suficientes para la inteligencia. **John Searle** llama esta creencia **Inteligencia Artificial Fuerte** y sostiene que es fundamentalmente equivocada. La tesis central de la inteligencia artificial fuerte es que los procesos realizados por una computadora son idénticos a los que realiza el cerebro, y por lo tanto se puede deducir que, si el cerebro genera conciencia, también las computadoras deben ser conscientes. En su opinión, las aplicaciones de IA tratan únicamente con la sintaxis, no con la semántica, por lo que un computador no puede explicar la intencionalidad o significado de lo que está procesando. La IA fuerte actualmente cae dentro de la ciencia ficción y la especulación futurista. El campo de la robótica es uno de los que más ha impulsado el estudio de la **inteligencia general** (**Inteligencia Artificial Fuerte**[49]). Si bien se han logrado importantes avances, también se han evidenciado las dificultades de tratar de generar modelos de procesos mentales que todavía los seres humanos no han logrado entender a un nivel suficiente como para poder describirlos, representarlos y modelarlos.

Desde la perspectiva tecnológica, en cambio los sistemas inteligentes no necesariamente deben emular los procesos característicos de los seres inteligentes. A esta premisa se la denomina **Inteligencia Artificial Débil**.

---

[49] http://en.wikipedia.org/wiki/Strong_AI

En el siguiente gráfico se presenta un mapa mental de los principales tópicos que, desde esta perspectiva, trata actualmente la inteligencia artificial:

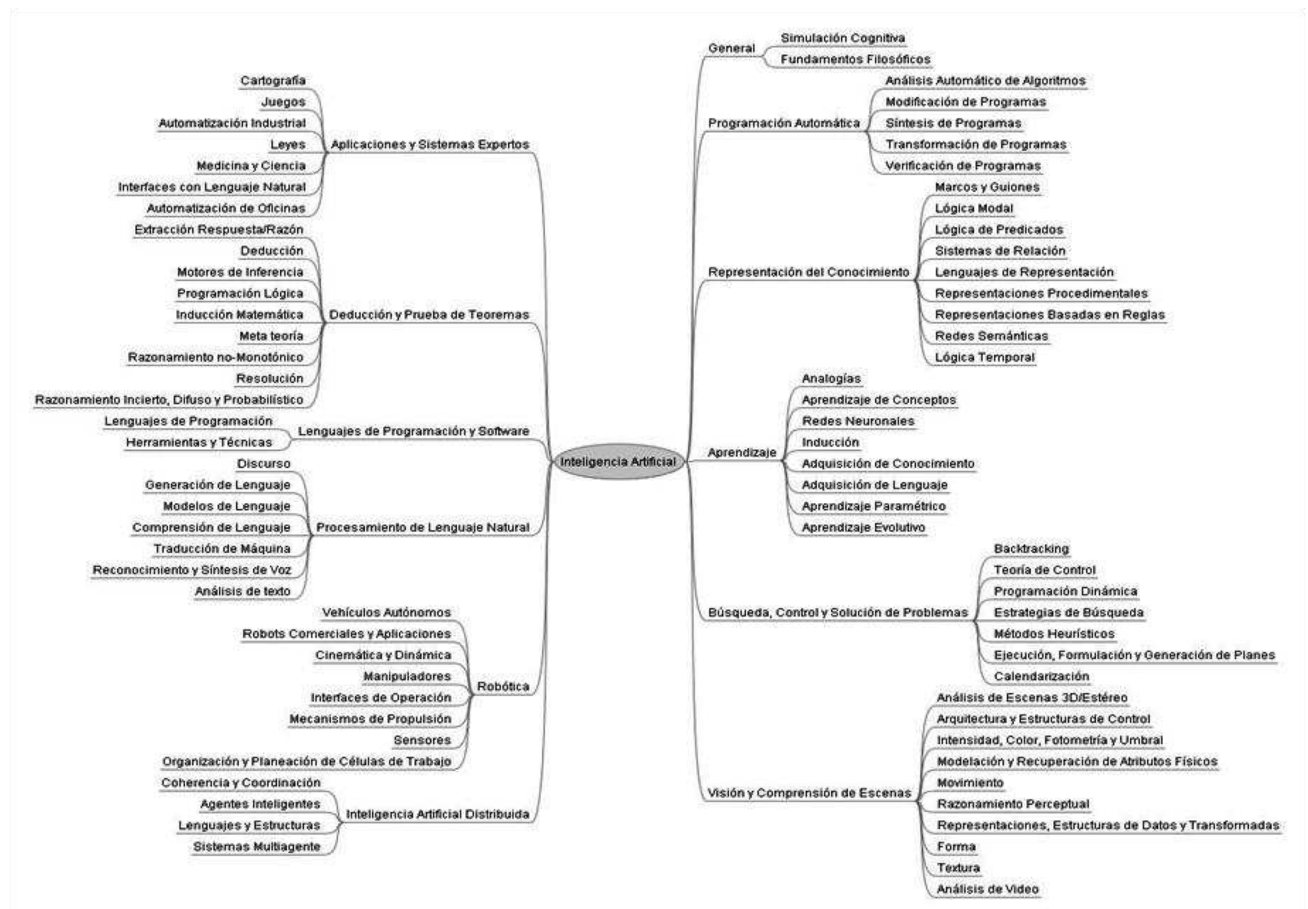


Figura 7. Tópicos relacionados con la Inteligencia Artificial

La actual proliferación y desarrollo de la infraestructura de comunicación basada en las redes como la World Wide Web (WWW), ha creado un escenario sin precedentes y está potenciando la probabilidad de éxito de los sistemas inteligentes para la recopilación masiva de información y su uso en la automatización de diversas funciones empresariales.

1.4.1 Ciencias de Computación

Las ciencias de la computación tienen su raíz en ingeniería eléctrica, matemática y lingüística. Sin embargo, en el último tercio del siglo XX, emergió como una disciplina científica distinta y desarrolló sus propios métodos y terminología.

Las Ciencias de Computación, pueden definirse como:

> ***Estudio de la teoría, experimentación e ingeniería que forman la base para el diseño y uso de computadores.***
>
> ***Incluye aquella parte de la informática que es posible automatizar a partir de sistemas de computación.***

Las ciencias de la computación comprenden una amplia variedad de tópicos, que van desde el análisis abstracto de algoritmos, teoría de la computación y gramáticas formales, hasta temas como lenguajes de programación, software y hardware.

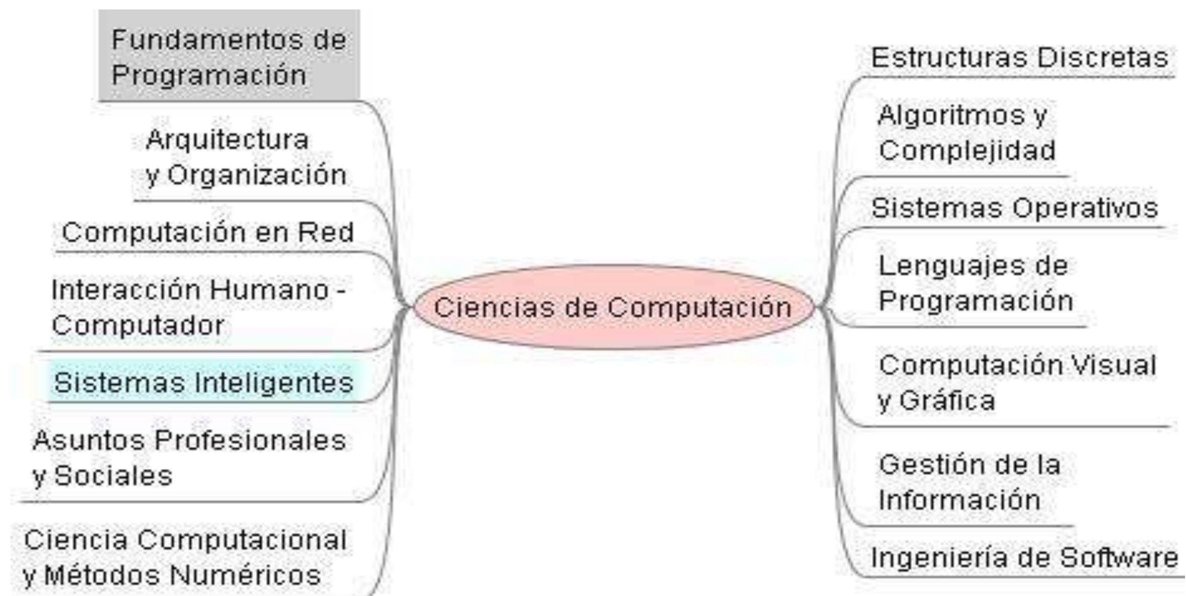


Figura 8. Tópicos relacionados con las Ciencias de Computación

1.4.2 Los Sistemas Inteligentes

Son parte de las ciencias de computación, que cubren una serie de tópicos teóricos – experimentales que sirven de base para la ingeniería de aplicaciones inteligentes. Esto se muestra en el siguiente gráfico:



Figura 9. Tópicos relacionados con los Sistemas Inteligentes

1.5 Presente y Futuro de los Sistemas Inteligentes

En un campo de investigación tan amplio y relativamente joven como la Inteligencia Artificial, no es raro que prevalezcan muchas controversias, tanto científicas como filosóficas. Según lo señala **Margaret Goden**[50], actualmente, "*la mayoría de investigadores en IA utilizan ya sea el modelo clásico o el modelo conexionista. Sin embargo, debido a que ambos modelos tienen debilidades y fortalezas complementarias, hay un creciente interés en los modelos híbridos que tratan de obtener lo mejor de ambos mundos*".

La IA a inicios del Siglo XXI es intrigantemente diversa. Varias apuestas científicas se están realizando y no es obvio si alguna propuesta híbrida eventualmente resultará ganadora. Además, a medida que avance la investigación, nuevos conceptos de IA, surgirán sin lugar a dudas. Por otro lado, tampoco está claro cómo las dificultades filosóficas afectarán la investigación científica de IA, en términos de su relación con la sociedad y la naturaleza.

En este escenario, resulta muy ilustrativo referirse a un artículo escrito por Stephen Cass y publicado en *Technology Review*, una revista del MIT[51], en el que se recogen opiniones de varios expertos en IA, que se citan a continuación:

- "*Usted podría preguntarse por qué no hubo algún robot que pudo enviarse a reparar los reactores japoneses*," dijo **Marvin Minsky**, que fue pionero en redes neuronales en la década de 1950 y contribuyó con avances significativos a los principios de IA y robótica. "*La respuesta es que hubo muchos progresos en la década de 1960 y 1970. Luego, algo se hizo mal. [Hoy] encontrará estudiantes entusiasmados con robots que juegan baloncesto o fútbol o bailan y le hacen caras graciosas. [Pero] no los están haciendo más inteligentes*".
- **Patrick Winston**, director del laboratorio de Inteligencia Artificial del MIT desde 1972 hasta 1997, culpó del estancamiento en parte a la disminución de fondos después del final de la guerra fría y a los intentos tempranos para comercializar la IA. Pero el mayor culpable, dijo, fue la "balcanización mecanicista" del campo de IA, con la investigación enfocada cada vez en especialidades más estrechas tales como redes neuronales o algoritmos genéticos. "*Cuando se dedican las conferencias a mecanismos, hay una tendencia a no trabajar en problemas fundamentales, sino más bien [sólo] en los problemas que se relacionan a los mecanismos*". Winston dijo que él cree que los investigadores en su lugar deberían centrarse en identificar esas cosas que hacen a los seres humanos distintos de otros primates, e incluso en lo que les hizo distintos de los Neandertal. Una vez los investigadores consideren que han identificado las cosas que hacen únicos a los seres humanos, dijo, deben desarrollar modelos computacionales de estas propiedades, implementarlas en sistemas reales para que puedan descubrir las brechas en sus modelos y les refinen según sea necesario. Winston especuló que el

---

[50] Margaret A. Boden. AI's Half-Century. AI Magazine 16(4): Winter 1995, 96-99
[51] Stephen Cass. *Unthinking Machines*. Technology Review, Published by MIT, Boston, MA, USA, May 4, 2011

ingrediente mágico que hace único al ser humano es nuestra capacidad de crear y entender historias utilizando las facultades que apoya el lenguaje: "*Una vez que se tengan historias, se tiene el tipo de creatividad que hace la especie diferente a cualquier otra*".

- Los dos lingüistas en el panel, **Noam Chomsky** y **Barbara Partee**, ambos hicieron contribuciones a la comprensión del lenguaje al considerarlo como un fenómeno computacional, en lugar de algo meramente cultural. Ambos consideraron que, entender el lenguaje humano podría ser la clave para crear máquinas realmente pensantes. "S*aber semántica es un requisito previo para que algo pueda llamarse inteligente*," dijo **Partee**. Por otra parte, **Chomsky** se mostró en desacuerdo con investigadores en aprendizaje de máquina que utilizan métodos puramente estadísticos para producir comportamiento que imita algo en el mundo, sin tratar de comprender el significado de ese comportamiento. Chomsky comparó a tales investigadores con científicos que estudian la danza de una abeja al retomar a su colmena, y producen una simulación basada estadísticamente en tal danza, sin intentar comprender por qué la abeja se comporta de esa manera. "*Es una noción muy novedosa de éxito [científico]. No conozco nada igual en la historia de la ciencia*," dijo **Chomsky**.

2 CAPÍTULO II: CEREBRO E INTELIGENCIA

2.1 Introducción

En esta unidad se presentan los fundamentos de la inteligencia, partiendo de la evolución del cerebro humano.

La familiarización y el análisis de los conceptos, hechos y principios asociados a estos tópicos, configuran el marco referencial para sustentar la comprensión de las funciones, procesos y atributos de los sistemas inteligentes tanto naturales, como artificiales.

2.2 Evolución del Cerebro

Las funciones superiores del pensamiento humano residen en la arrugada capa exterior del cerebro conocida como corteza cerebral. Esta región del cerebro humano creció a partir de una zona dedicada al análisis de los olores en el cerebro del pequeño mamífero de los bosques que vivió hace 100 millones de años.

Los avances de la investigación en la neurociencia y las ciencias cognitivas han permitido ir desvelando las maravillas del *telar mágico*.

Cada estrella, cada planeta y cada criatura viviente, empezaron a existir como resultado de una serie de acontecimientos, que fueron puestos en marcha en el instante en que se produjo la explosión cósmica (Big Bang), hace unos 20.000 millones de años. El sistema solar existe desde hace unos 4.600 millones de años.

Por los restos fósiles, se sabe que algunos organismos relativamente complicados, como las bacterias, existían ya cuando la Tierra tenía tan sólo 1.000 millones de años de edad.

Hace 1.000 millones de años, tras unos 3.000 millones de años de invisible progreso de las bacterias, se produjo un acontecimiento importante, aparecieron sobre la Tierra las primeras criaturas multicelulares.

Hace 600 millones de años aparecen las primeras criaturas de cuerpo duro. Eran los antepasados de la almeja, la estrella de mar, la langosta, los insectos.

En los siguientes 100 millones de años, algunos animales de cuerpo duro evolucionaron a una forma de criaturas vertebradas.

Luego, hace unos 450 millones de años, aparecen los primeros peces, que aparte de su esqueleto, poseían un cerebro. Muy pequeño, pero el primero que aparecía en la Tierra.

2.2.1 La Conquista de Tierra Firme

Las plantas fueron las primeras en abrirse camino hacia la tierra firme, luego, siguieron los insectos, atraídos por la abundancia de vegetación.

Una variedad de pez, denominado **crosopterigio**, realizó la hazaña de salir del agua a conquistar la tierra, hace unos 370 millones de años. El estudio del esqueleto de las aletas pares de los peces llamados crosopterigios presenta homologías con el esqueleto de los miembros de los vertebrados terrestres (o tetrápodos). El único superviviente de los crosopterigios, es el **celacanto**[52]. Este es un pez de aletas lobuladas que apareció a finales del Devónico. A menudo se le llama fósil viviente porque tiene numerosas peculiaridades que sólo se encuentran en peces fósiles. Aunque se creía extinto desde hace mucho tiempo, el celacanto fue redescubierto en 1938 en África del Sur.

Se han descubierto piezas óseas que corresponden al húmero, el radio y el cúbito y que no tienen equivalentes en otros peces. Este resultado de la anatomía comparada permite considerar a los crosopterigios como ancestros muy probables de los tetrápodos. Sin embargo, recientes análisis de los ácidos nucleídos de peces y anfibios actuales, pusieron dudas sobre estos resultados ampliamente aceptados. Ahora se propone a los **dipnoos**[53,54] como los lejanos ancestros de los mamíferos.

2.2.2 Del Pez al Mamífero

Los descendientes de los dipnoos, se establecieron firmemente en las orillas de ríos y arroyos; y, gradualmente fueron evolucionando. Primero aparecieron los anfibios (antepasados de ranas y sapos); y, luego, hace 300 millones de años, dieron lugar al aparecimiento de los reptiles. Estos florecieron y dieron grandes líneas de evolución: lagartos y serpientes, tortugas, cocodrilos, dinosaurios, pájaros y mamíferos.

Los antepasados de los mamíferos fueron una forma transitoria llamada **terápsidos**[55]. Los terápsidos fueron miembros de un grupo de reptiles a partir de los cuales evolucionaron, con el tiempo, los mamíferos. Los terápsidos vivieron desde finales del paleozoico hasta principios del mesozoico. Alcanzaron su apogeo hace unos 250 millones de años y durante los siguientes 50 millones de años dominaron en tierra firme, después comenzaron a extinguirse. Los terápsidos fueron animales terrestres de cuatro patas, caracterizados por:

- Articulaciones dobladas paralelamente al cuerpo. Cuerpo alzado sobre el suelo.
- Metabolismo de sangre caliente
- Rasgos rudimentarios de amor y cuidados maternos.

---

52 http://es.wikipedia.org/wiki/Coelacanthimorpha
53 http://www.cienciahoy.org.ar/hoy48/nbanba01.htm
54 http://forum-psicologos.blogspot.com/2010/08/dipnoo.html
55 http://www.telefonica.net/web2/paleontologiaernesto/Vertebrados/MTerrestres/Terapsidos.html

Ciertos terápsidos tenían el tamaño de pequeños roedores, otros eran tan grandes como los actuales hipopótamos. Algunos eran herbívoros y otros carnívoros. Presentaban determinados atributos típicos de los mamíferos; por ejemplo, entre los dientes de algunas especies se diferenciaban incisivos cortantes, caninos desgarradores y molares trituradores.

La evolución de los reptiles que condujo al dinosaurio empezó hace unos 225 millones de años, durante el apogeo de los terápsidos. Los gigantes terrestres fueron producto de un período de más de 100 millones de años de clima suave y comida abundante

Los terápsidos no tenían forma de competir con los dinosaurios carnívoros. Hace unos 180 millones de años los descendientes de los terápsidos se redujeron al tamaño de ratones, para poder sobrevivir, pero ya tenían rasgos de auténticos mamíferos.

2.2.3 Mamíferos Primitivos

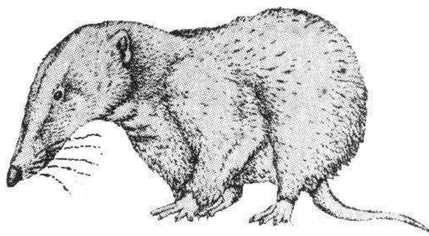


Figura 10. Megazostrodón (mamífero primitivo)

El **megazostrodón**[56], fue un género de mamíferos primitivos parecidos a la musaraña que vivieron principalmente en el jurásico (hace 208 a 145 millones de años) en Lesoto, en el sur de África. Fueron animales pequeños, activos, de sangre caliente, probablemente provistos de pelo, que gestaban a sus crías en su interior. Tenían ojos muy pequeños, nariz y oídos desarrollados, sugiriendo que eran animales nocturnos. Gracias a sus cerebros relativamente grandes, eran más inteligentes que cualquier otra criatura de aquel entonces.

2.3 La Evolución de la Inteligencia

La historia de la evolución de la inteligencia es la historia de la evolución del cerebro de los mamíferos. Hace 65 millones años, después de la extinción de los dinosaurios, evolucionaron los modernos mamíferos y el cerebro se expandió para el control neuromuscular y la integración de los diferentes sentidos.

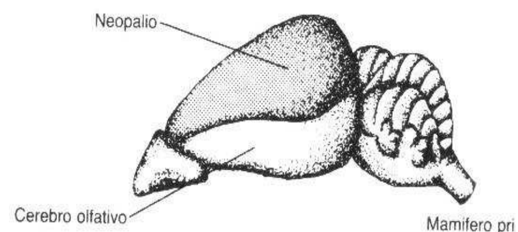


Figura 11. Cerebro de un mamífero primitivo

Investigadores, encabezados por **Timothy B. Rowe**[57], paleontólogo de la Universidad de Texas, publicaron un reporte en la revista ***Science***, en el que sugieren que el tamaño

[56] http://es.wikipedia.org/wiki/Megazostrodon

[57] Timothy B. Rowe, Thomas E. Macrini, and Zhe-Xi Luo. ***Fossil Evidence on Origin of the Mammalian Brain***. Science 20 May 2011: Vol. 332 no. 6032 pp. 955-957

relativo de los cerebros de las primitivas criaturas se expandió a los niveles de los mamíferos de hoy, mediante una expansión de los bulbos olfativos, la neo corteza, la corteza olfativa (piriforme) y el cerebelo. Según los hallazgos de estos investigadores, el cerebro de los mamíferos evolucionó en tres pasos. En el primer paso se dio una mejora del sentido del olfato, en el segundo un aumento de los sentidos del tacto a través de pelo corporal y en el tercero una mejora de la coordinación neuromuscular, es decir, una mejora en la habilidad de producir movimientos musculares diestros usando los sentidos.

Una vez el cerebro de los mamíferos se hizo más grande y complejo fue permitiendo el uso de nuevos recursos, como la visión en color, la ecolocación de los cetáceos o incluso la habilidad de sentir el campo eléctrico del ornitorrinco. En los primates la evolución modificó lo que ya había en los cerebros y los expandió aún más. Finalmente esto dio lugar a los seres humanos. Los mamíferos tienen unas 10 veces más genes relacionados con el sentido del olfato que otros vertebrados, aunque en el ser humano muchos de ellos, relacionados con receptores de olor específicos, no son funcionales.[58]

2.3.1 El Cerebro Humano

El cerebro humano está formado por varias zonas diferentes que evolucionaron en distintas épocas. Un cerebro humano medio posee las siguientes características:

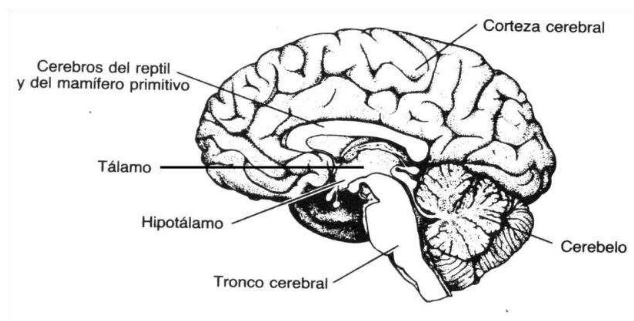


Figura 12. Regiones de un cerebro humano

- Pesa unos 1350 g.
- Ocupa un volumen de 1300 cm^3.
- El 90% de la masa que recubre el tronco cerebral, es la corteza cerebral, que es la región más nueva.
- Tiene 100.000 millones (10^{11}) de neuronas.
- Existen 100 billones de conexiones (10^{14}) entre ellas (1.000 conexiones por neurona), de las cuales el 70% está en la corteza cerebral.
- Su tamaño no ha cambiado significativamente en los pasados 100.000 años.

2.3.2 El Cerebro Triádico

Waldemar De Gregori describe el comportamiento del ser humano según el modelo de cerebro triádico[59]. Éste se identifica según tres procesos mentales: cerebro central o inteligencia operativa; cerebro derecho o inteligencia emocional y cerebro izquierdo o inteligencia lógica.



Figura 13. Cerebro Triádico

[58] http://neofronteras.com/?p=3504
[59] De Gregori, Waldemar. *Construcción del Poder de Los Tres Cerebros*. Bogotá: Kimpres, 1999.

El cerebro central se asocia con la acción, la experiencia, el pensamiento concreto, la conducta del ser; el cerebro izquierdo con la ciencia, la matemática y el saber; y el cerebro derecho con la intuición, la fe, las emotividades, el arte y la religión. Estos tres cerebros integran un sistema que se correlaciona, determina y explica el comportamiento humano.

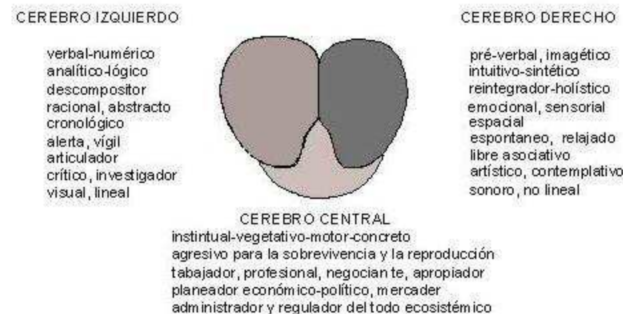


Figura 14. Funciones del cerebro triádico

El **Revelador del Cociente Mental Triádico** (**RCMT**)[60] es un modelo cuantitativo que, de manera proporcional, determina las inclinaciones del cerebro triádico, de acuerdo con las posiciones frente al mundo, los temores, las relaciones con los demás, los criterios para evaluar el arte y en general, su actuar en los contextos lógicos, conductuales y lúdicos. Del análisis del cerebro triádico pueden surgir estrategias para la pedagogía y la administración. De acuerdo con la **cibernética social**[61], éste organiza conocimientos, define jerarquías y crea visiones para sacar el mejor provecho de las debilidades y las oportunidades.

2.4 Inteligencia Humana

Ningún otro tema ha provocado tanta controversia como el estudio de la inteligencia humana. Uno de los estudios científicos más influyentes acerca de la inteligencia es aquel que se basa en pruebas psicométricas.

Los procesos cerebrales relacionados con la inteligencia son todavía muy poco comprendidos. Inteligencia, es un concepto integrador de muchas aptitudes y capacidades intelectuales:

- Aptitud para crear relaciones, ya sea de manera sensorial o intelectual.
- Habilidad para adquirir, comprender y aplicar *conocimiento*.
- Aptitud para recordar**,** pensar y razonar**.**
- Capacidad global de un individuo para pensar, actuar deliberadamente en forma racional, y relacionarse de manera eficaz con su entorno.

[60] http://es.scribd.com/doc/38547975/Revelador-Del-Cociente-Mental-Triadico
[61] http://pitagorastutores.blogspot.com/2010/10/cibernetica-social-paradigma.html

El ser humano posee en su inteligencia la facultad del despegue y de la liberación, en este sentido se aleja de la monotonía animal (rutinas biológicamente programadas).

No está como el animal pendiente del estímulo, sino que el ser humano elige el estímulo. Es capaz de decidir lo que quiere aprender. La atención inteligente consiste en atender a cosas que nada interesa.

Cada ser humano interpreta la realidad de acuerdo con sus proyectos, que son los activadores de significados. Esta capacidad permite pensar no sólo en cosas que existen, sino también descubrir o inventar otras posibilidades. La inteligencia humana no sólo trasfigura el sentido de las cosas, también se empeña en cambiarlas.

La función de la inteligencia no es sólo conocer, ni resolver ecuaciones diferenciales, ni jugar al ajedrez, sino dirigir el comportamiento para resolver con éxito problemas vitales, afectivos o profesionales, para saber elegir nuestras metas y poder realizarlas.

Pero, la medida del ser humano es la desmesura. La historia de la humanidad es una crónica de grandeza y a la vez de crueldad. La gran transfiguración de la inteligencia aparece cuando el ser humano es capaz de iniciar, controlar y dirigir sus operaciones mentales. La persona incapaz de controlar sus impulsos no es inteligente.

La inteligencia tiene que tratar con conocimiento y también con valores. La inteligencia ha inventado un modo de relacionarse con los valores. Los siente pero también los piensa.

> ***La ciencia, el arte y las demás ramas de la cultura sirven en la medida que ayuden a resolver problemas de la humanidad, a adecentar sus vidas, a construir el azaroso orbe de la dignidad humana. La inteligencia es nuestro gran recurso, nuestro gran riesgo y nuestra gran esperanza.*** [62]

2.5 Teoría Triádica de la Inteligencia

Robert J. Sternberg, catedrático e investigador de la universidad de Yale, en 1985 publicó una teoría a la que llamó **Teoría Triádica de la Inteligencia Humana**[63]. La teoría triádica describe la relación de la inteligencia con tres dimensiones de la persona, áreas a las que el autor denomina subteorías. Se describen a continuación:

- **La Subteoría Componencial** tiene que ver con el mundo interno del individuo, con el pensamiento analítico y académico. Investiga, planea y ejecuta. Mecanismos mentales que articulan la conducta inteligente (mundo interno):

[62] Marina, José Antonio. *El Vuelo De La Inteligencia*. De Bolsillo, Barcelona, España, 2003
[63] Sternberg, R. J. *Beyond IQ: A Triarchic Theory of Intelligence*. Cambridge University Press, 1985

- Meta-componentes o procesos mentales superiores utilizados para la planeación, supervisión y evaluación de acciones.
- Componentes de ejecución o procesos utilizados para la ejecución de tareas.
- Componentes de adquisición del conocimiento, utilizados para conocer el significado de los conceptos, a partir del contexto.

- **La Subteoría Experiencial** explica su relación con el mundo externo, la forma en que maneja su experiencia en las situaciones cotidianas, su pensamiento creativo. Busca originalidad e innovación. Posibilidad de desarrollo de la inteligencia por medio de los componentes de adquisición del conocimiento (mundo interno – externo):
  - Identificación y comprensión de conceptos que guardan correspondencia con la información del contexto. Incluye intuición y perspicacia ante situaciones novedosas a las que se enfrenta una persona.
  - Control y automatización del procesamiento de la información. Conocimiento procedimental o habilidades motoras que, por medio de la constancia de la práctica, suelen ejecutarse con mayor rapidez y precisión.
- **La Subteoría Contextual** hace referencia a la forma en que el individuo se mueve en su entorno, al pensamiento práctico (*street-smart*), adaptativo y exitoso. Implica la solución de problemas. El ser humano manifiesta diferentes conductas dependientes de la naturaleza de la tarea, de su conocimiento, su sentimiento y de las condiciones del entorno (mundo externo):
  - Adaptación al entorno
  - Selección de un nuevo entorno
  - Transformación del entorno

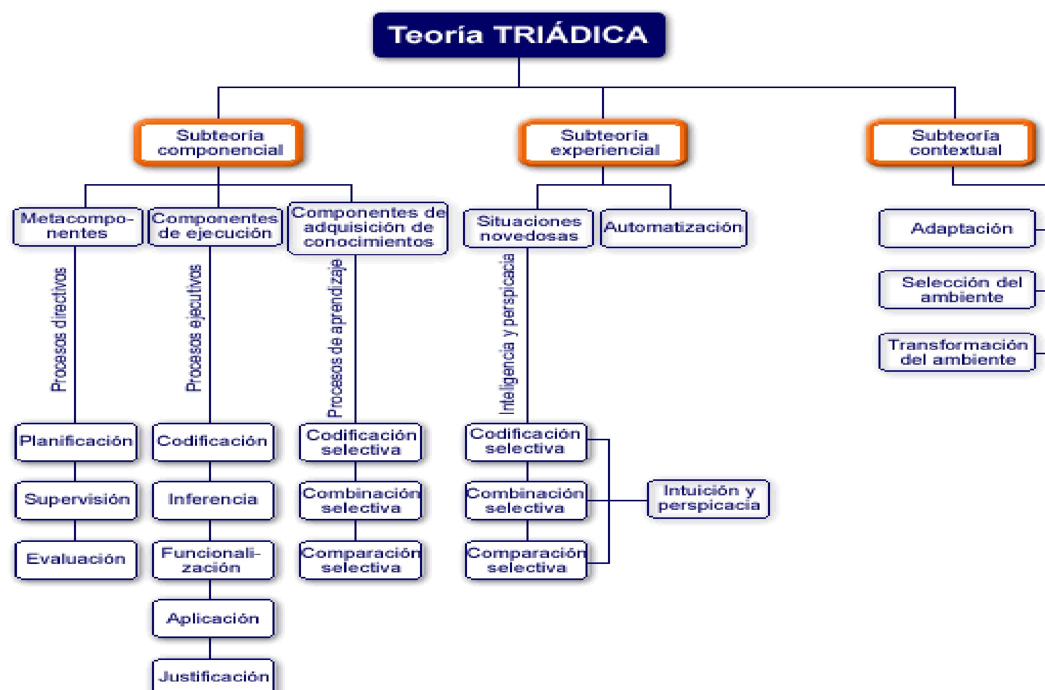


Figura 15. Componentes de la Teoría Triádica de la Inteligencia

2.6 Teoría de las Inteligencias Múltiples

Las últimas teorías en psicología sobre la multiplicidad de las inteligencias, elaboradas por el profesor **Howard Gardner** y sus colaboradores del proyecto Zero de la Escuela Superior de Educación de Harvard, dejan atrás la concepción casi única de la inteligencia[64]. La **Teoría de las Inteligencias Múltiples** entiende la competencia cognitiva como un conjunto de habilidades, talentos y capacidades mentales que llama inteligencias. Todas las personas poseen estas habilidades, capacidades mentales y talentos en distintos niveles de desarrollo. Su aplicación permite lograr[65]:

- El incremento de la autoestima.
- La minimización de los problemas de conducta.
- El desarrollo de las habilidades de cooperación y de liderazgo.
- El incremento del interés y de la dedicación al aprendizaje.
- El incremento del conocimiento.
- La presencia permanente del buen humor.

2.6.1 Inteligencia Intrapersonal

Capacidad de construir una percepción precisa de sí mismo y de organizar y dirigir su propia vida. Incluye la autodisciplina, auto comprensión y autoestima. La evidencian quienes tienen:

- Actitudes reflexivas y de ayuda a los demás.
- Capacidad para aprender a trabajar solos, a su propio ritmo.
- Precisión para valorar sus puntos fuertes y débiles.

2.6.2 Inteligencia Interpersonal

Disposición para entender e interactuar eficazmente con los demás. Incluye la sensibilidad a expresiones faciales, voz, gestos y posturas; y, la habilidad para responder. La evidencian quienes:

- Tienen habilidad para trabajar en grupo y logran potenciar a los demás.
- Aprenden mejor compartiendo, comparando, relacionando.
- Se destacan interactuando con la gente, liderando, organizando, comunicando, resolviendo conflictos, negociando con éxito.

2.6.3 Inteligencia Naturalista

Capacidad de distinguir, valorar y trabajar con animales, plantas y más componentes del medio ambiente. Incluye las habilidades de observación y experimentación del entorno. La evidencian quienes:

---

[64] Gardner, Howard. *Frames of Mind: The Theory of Multiple Intelligences.* New York: Basic Books, 1983

[65] http://quizfarm.com/quizzes/Inteligencias+Multiples/profesorrod/test-de-inteligencias-multiples/

- Aman la naturaleza, los animales, las plantas y en general, el mundo natural.
- Se destacan y disfrutan trabajando en la naturaleza, identificando y protegiendo la flora, la fauna y su entorno natural.

2.6.4 Inteligencia Visual-Espacial

Capacidad para pensar en tres dimensiones. La evidencian quienes:

- Trabajan en diseño, esquemas, mapas conceptuales y mentales.
- Aprenden mejor con dibujos y colores, visualizando, usando su "ojo mental".
- Se destacan leyendo e interpretando mapas, gráficos, imaginando, resolviendo rompecabezas.

2.6.5 Inteligencia Lógico-Matemática

Capacidad para razonar y manejar con destreza los números. Incluye la sensibilidad a los esquemas y relaciones lógicas, las afirmaciones y las proposiciones, las funciones y otras abstracciones relacionadas. La evidencian quienes:

- Se destacan en matemáticas y lógica.
- Analizan y resuelven con facilidad problemas.
- Afrontan con entusiasmo cálculos numéricos, estadísticas y presupuestos.

2.6.6 Inteligencia Musical

Capacidad de percibir, transformar y expresar formas musicales. Incluye la sensibilidad al ritmo, tono y timbre. La evidencian quienes:

- Se sienten atraídos por los sonidos de la naturaleza y por todo tipo de melodías.
- Disfrutan siguiendo el compás con el pie, golpeando algún objeto rítmicamente.
- Aprenden mejor a través del ritmo, la melodía, el canto, oyendo música.
- Se destacan en canto, reconociendo sonidos y melodías, tocando instrumentos musicales.

2.6.7 Inteligencia Verbal-Lingüística

Capacidad de usar las palabras de manera efectiva, en forma oral y escrita. Incluye la habilidad en el uso de la sintaxis, fonética, semántica y la aplicación práctica del lenguaje. La evidencian quienes:

- Gustan redactar historias, leer, jugar con rimas, trabalenguas y aprenden con facilidad otros idiomas.
- Aprenden leyendo, hablando, escribiendo, discutiendo y debatiendo.

2.6.8 Inteligencia Corporal - Kinestésica

Medida para expresar ideas y sentimientos con el cuerpo. Destreza con las manos para transformar elementos. La evidencian quienes:

- Tienen habilidades de coordinación, equilibrio, flexibilidad, fuerza y velocidad, así como percepción de medidas y volúmenes.
- Se destacan en cirugía, danza, deportes, arte y habilidad manual, utilización de herramientas y ejecución de instrumentos musicales.

Según la teoría propuesta por Gardner, todos los seres humanos poseen las ocho inteligencias en mayor o menor medida. Al igual que con los estilos de aprendizaje no hay tipos puros, y si los hubiera les resultaría imposible funcionar. Un ingeniero civil necesita una inteligencia espacial bien desarrollada, pero también necesita de todas las demás, de la inteligencia lógico-matemática para poder realizar cálculos de estructuras, de la inteligencia interpersonal para poder presentar sus proyectos, de la inteligencia corporal-kinestésica para poder conducir su coche hasta la obra, etc.

2.6.9 Inteligencia Emocional

En 1995, **Daniel Goleman** propone la **Inteligencia Emocional** que complementa al modelo de inteligencias múltiples propuesto por **Howard Gardner**[66]. Goleman, en su libro, estima que la inteligencia emocional se puede organizar en cinco capacidades: conocer las emociones y sentimientos propios, manejarlos, reconocerlos, crear la propia motivación, y gestionar las relaciones. En esencia, la inteligencia emocional no es sino la capacidad de relación positiva con los demás; la demostración de una notable autoestima y habilidad para valorar a las personas; la posesión de fortaleza y ánimo para superar las dificultades, sin dejarse abatir por los fracasos. La demuestran quienes tienen[67]:

- Buen sentido del humor y habilidad para mantener relaciones sociales.
- Curiosidad creativa y afán de innovación.
- Capacidad para resolver problemas y para afrontar con buen ánimo los retos de la vida diaria.

---

[66] Goleman, Daniel. *Emotional Intelligence*. New York: Bantam Books. 1995
[67] http://www.helios3000.net/tests/eq.shtml

3 CAPÍTULO III: CONOCIMIENTO Y APRENDIZAJE

3.1 Introducción

En esta unidad se presentan los fundamentos del conocimiento y el aprendizaje. La familiarización y el análisis de los conceptos, hechos y principios asociados a estos tópicos, configuran el marco referencial para sustentar la comprensión de las funciones, procesos y atributos de los sistemas inteligentes tanto naturales, como artificiales.

3.2 Teoría de la Complejidad

En el proceso de observación inscrito en los fundamentos de la **ciencia clásica** se considera al sujeto cognoscente como un observador neutro, separado del objeto y despojado de sus emociones, intereses y creencias.

Desde este paradigma, el objeto por conocer, es entendido de forma múltiple y diversa (**multidisciplinar**), como las disciplinas que lo abordan, y aún más, en el interior de cada disciplina se presentan divergencias para entenderlo.

La **complejidad** surgió, a fines de los años 60, a partir de la **Teoría de la Información**, la **Cibernética** y la **Teoría de Sistemas**. Importantes avances en los más diversos dominios científicos en las últimas décadas, han puesto en evidencia que la realidad lejos de ser un simple agregado de partes en el que se puede ir añadiendo o quitando elementos para explicar sus características y construir totalidades ordenadas y predecibles, es más bien una trama de relaciones en dinámica permanente, regida por la incertidumbre y el caos.

Efectivamente, el saber contemporáneo se presenta al observador como una torre de babel, cuya multitud de objetos, métodos, posiciones, perspectivas, teorías, concepciones sobre el mundo y lenguas en las que se describen unos y otras, convierten el panorama en algo complejo.

Edgar Morin[68], sociólogo y filósofo francés, sostiene que el pensamiento complejo está basado en dos aspectos clave:

- Integra todos los elementos que puedan aportar orden, claridad, distinción, precisión en el conocimiento, pero rechaza las consecuencias mutilantes, reduccionistas, unidimensionalizantes que puede producir una simplificación que oculte todos los vínculos, las interactuaciones, las interferencias que hay en el mundo real.

---

[68] http://es.wikipedia.org/wiki/Edgar_Morin

- Reconoce que el conocimiento completo es imposible. En este sentido se formula uno de los axiomas de la complejidad: la imposibilidad, incluso teórica, de una omniciencia. Esto implica la afirmación de un principio de incompletitud y de incertidumbre.

El pensamiento de **Edgar Morin** conduce a un modo de construcción que aborda el conocimiento como un proceso **inter** y **trans disciplinar** que es a la vez, biológico, cerebral, espiritual, lógico, lingüístico, cultural, social e histórico, mientras que la **epistemología tradicional** asume el conocimiento sólo desde el punto de vista **cognitivo**. Este nuevo planteamiento tiene enormes consecuencias en el planteamiento de las ciencias, la educación, la cultura, la sociedad.[69]

> ***El conocimiento humano es la traducción de la percepción del mundo real a partir de los símbolos, de la percepción de los discursos y teorías que los seres humanos en sus conversaciones sobre el mundo y sus estados, infieren y/o conciben como eventos, leyes, fenómenos, sistemas, etc. Procesos que implican computaciones y cogitaciones mediatizadas por las informaciones, representaciones y expectativas que la vida en sus quehaceres conlleva, posibilita y obstaculiza.***[70]

3.2.1 Epistemología de la Complejidad

Plantea que el conocimiento solo es posible si se considera al sujeto como elemento activo del proceso de conocer.

- El sujeto construye el conocimiento.
- El conocimiento está asociado a la vida del sujeto.
- Conocimiento, emoción y actividad son inseparables.

Para la complejidad, el objeto del conocimiento está construido por el sujeto desde sus determinaciones emocionales, estructurales y en relación con su realidad, es decir su mundo o su contexto.

Junto a esta noción de dominios psicológicos, de corte individualista, otros autores destacan el carácter de construcción compartida del conocimiento cotidiano en dominios culturales o pragmáticos[71,72,73]. Desde esta perspectiva, los procesos de Aprendizaje Social constituyen medios idóneos para asegurar la apropiación del conocimiento y su transformación en resultados pragmáticos. Construir un dominio cultural implica elaborar

---

[69] Morin, Edgar. *Los 7 saberes necesarios para la educación del futuro*. UNESCO, Francia, 1999.

[70] Morin, Edgar. *Introducción al pensamiento complejo*. Barcelona, Gedisa Editorial, 1998

[71] Levine, J. M., Resnick, L. B, and Higgins, E. T. *Social foundations of cognition*. Annual Review of Psychology 44, 1993, pp. 585-612.

[72] Greeno, J. G. *Number sense as situated knowing in a conceptual domain*. Journal for Research in Mathematics Education 22 (13), 1991, pp. 170-218.

[73] Lave, J., & Wenger, E. *Situated Learning: Legitimate Periperal Participation*. Cambridge University Press, UK, 1990.

representaciones mentales sobre éste y además, saberlas usar y poder participar con ellas en una comunidad de práctica, en donde el desarrollo cultural y el logro del bienestar social juegan el papel central.

3.2.2 Ontología de la Complejidad

Entiende que la realidad lejos de ser algo ordenado que se puede predecir, es una trama de relaciones que operan bajo la dinámica de la no linealidad, de la recursividad organizacional, o la **autopoiesis**[74].

La complejidad, además plantea que no existe "la realidad" sino múltiples realidades en las que a partir del aparecimiento de los primates, se genera un nuevo nivel de realidad, o realidad intersubjetiva[75].

Desde el punto de vista de la **epistemología evolutiva** la realidad intersubjetiva se define como una condición en la cual uno puede conocerse, solamente en relación con los otros, en la que no solamente están presentes los aspectos de protección afectiva recíproca, sino que es al mismo tiempo una cuestión de individualización. Cada miembro del grupo puede conocerse y se conoce a sí mismo en relación a la realidad alrededor de él, en relación con los otros, viéndose en los otros. La realidad intersubjetiva es una realidad afectiva y de conocimiento. En los primates, se ve que el conocimiento es siempre interactivo con los otros.

3.2.3 Axiología de la Complejidad

El conocimiento establece inevitablemente un determinado grado de compromiso, pues conocer, hacer y vivir no son cosas separables.

La realidad del ambiente, nuestra identidad transitoria y supervivencia mutua están asociadas en un juego constructivista. La bioética propone una ciencia no neutral, es decir una ciencia con conciencia.

3.2.4 Metodología de la Complejidad

Señala la necesidad de introducir lógicas polivalentes, es decir que superen a la lógica binaria Aristotélica y se pueda dar cuenta de la realidad difusa, incierta, y hasta caótica del mundo real.

Desde esta perspectiva, la **metodología** debe tener **enfoque cualitativo** que guíe el **análisis cuantitativo**, acepte los enfoques **inter y trans disciplinares** e introduzca el **análisis contextual** para explicar el conocimiento.

---

[74] Varela F J; Maturana H R & Uribe R. *Autopoiesis: the organization of living systems, its characterization and a model*. Biosystems 5, 1974, pp. 187–196.

[75] Guidano V F. *De la revolución cognitiva a la intervención sistémica en términos de complejidad: La relación entre teoría y práctica en la evolución de un terapeuta cognitivo*. Revista de Psicoterapia 10,1990, pp. 113-129.

3.3 Conocimiento

Desde la perspectiva de la complejidad, el conocimiento no es global sino que está diversificado en dominios ligados a entornos pragmáticos de actividades y prácticas culturales. En estos entornos prácticos, propiciados por las comunidades de práctica, las personas desarrollan unas habilidades específicas mientras realizan determinadas actividades culturales. Como resultado de esta práctica regulada, van recolectando sus experiencias de dominio que servirán de base para la construcción de su conocimiento cotidiano.

En consecuencia, el conocimiento tiene diferentes significados, dependiendo del contexto, pero por regla general está asociado con: significado, instrucción, valores, información, comunicación, representación, aprendizaje, estímulo mental y satisfacción personal.

3.3.1 Datos, Información y Conocimiento

Los datos consisten de hechos o cifras obtenidos a partir de experimentos, estudios, observaciones o investigaciones. Los datos son los elementos primitivos que conllevan a la información, cuando al ser procesados en un determinado contexto, transmiten algún significado a los usuarios. Esto es, permiten al usuario comprender ciertas relaciones inmersas en los datos procesados.

El análisis de la información permite llegar al conocimiento, a través de la comprensión de patrones asociados con el lenguaje, hechos, conceptos, principios, procedimientos, reglas, ideas, abstracciones, lugares, costumbres y asociaciones. El conocimiento se sustenta si desarrolla habilidad para utilizar estas nociones en la comprensión y modelaje de los diferentes aspectos del ambiente en el que se realizan las actividades cotidianas.

A su vez, el logro de mayores destrezas para aprender y hacer cada vez mejor las cosas, genera talento. La inteligencia, el talento y el conocimiento están íntimamente ligados. Se puede decir que la inteligencia requiere de la posesión y acceso al conocimiento, para desarrollar el talento y lograr alcanzar la sabiduría, el estado en el que se logra la comprensión de los principios subyacentes en el mundo que nos rodea.



Figura 16. De los datos a la sabiduría

En esencia, se puede decir que el conocimiento es un cambio en la conducta que se adquiere, generalmente, a través de la práctica o la experiencia.

3.4 Aprendizaje

El aprendizaje es algo que está íntimamente asociado a nuestras vidas. Lo realizamos de manera consciente o inconsciente. Los elementos del aprendizaje, pueden ser hechos explícitos o el tipo de aprendizaje experimental que viene de la constante repetición. En este contexto, la educación es un intento consciente de promover el aprendizaje.

A partir de las recomendaciones dadas por la Comisión Internacional de UNESCO, sobre Educación para el Siglo XXI[76], la educación debe estructurarse en torno a aprendizajes fundamentales que, en el transcurso de la vida, serán para cada persona los pilares del conocimiento:

- El **saber ser** (actitudes, valores, dominio afectivo);
- El **saber vivir** (actitud positiva frente a la vida, construyendo su propio futuro);
- El **saber convivir** con otros (saber hacer con otros, trabajar en equipo, trabajo colaborativo, habilidades interpersonales, dominio relacional o social);
- El **saber conocer** o dominio cognoscitivo, aprender a aprender;
- El **saber hacer** (habilidades, destrezas, tanto intelectuales como del dominio psicomotor);
- El **saber servir** (integrar tanto lo que sabe, lo que es, lo que sabe hacer, asumiendo un compromiso social-profesional con la construcción de un mundo mejor, dominio de comportamiento ético); y,
- El **saber emprender** (aprender a identificar oportunidades donde otros ven amenazas).

Para promover y lograr estos aprendizajes fundamentales, los teóricos del aprendizaje han desarrollado diversas tipologías que distinguen al aprendizaje, de acuerdo con diferencias en lo que se está aprendiendo. Entre las conocidas están las taxonomías de objetivos educativos en el dominio cognoscitivo publicado por Bloom[77]; la tipología del aprendizaje publicada por Gagné y Briggs[78]; la taxonomía del dominio afectivo por Krathwohl, Bloom y Masia[79]; y, la del dominio psicomotor desarrollada por Harrow[80].

---

[76] http://unesdoc.unesco.org/images/0010/001095/109590so.pdf

[77] Bloom B. S. *Taxonomy of Educational Objectives, Handbook I: The Cognitive Domain.* New York: David McKay Co Inc.1956.

[78] Gagné, Robert & Briggs, Leslie. *Principles of Instructional Design, 2ª. Ed.* Holt, Rinehart y Winston, USA, 1979.

[79] Krathwohl, David R., Benjamin S. Bloom, and Bertram B. Masia. *Taxonomy of educational objectives, Book 2: Affective domain*. New York: Longman, 1964

[80] Harrow, A. *A Taxonomy of Psychomotor Domain: A Guide for Developing Behavioral Objectives*. New York: David McKay.

Benjamin Bloom, propone el desarrollo de tres dominios psicológicos principales: cognitivo, afectivo y psicomotor, los que se complementan con el dominio relacional o social.

3.4.1 Dominio Cognitivo

Hace referencia al aprender a conocer. Es el tipo de aprendizaje, que tiende a la adquisición de conocimientos clasificados y codificados; y al dominio de los instrumentos mismos del saber. Puede considerarse a la vez medio y finalidad de la vida humana. Aprender para conocer supone, en primer término, aprender a aprender, ejercitando la atención, la memoria y el pensamiento. El proceso de adquisición del conocimiento no concluye nunca y puede nutrirse de todo tipo de experiencias.

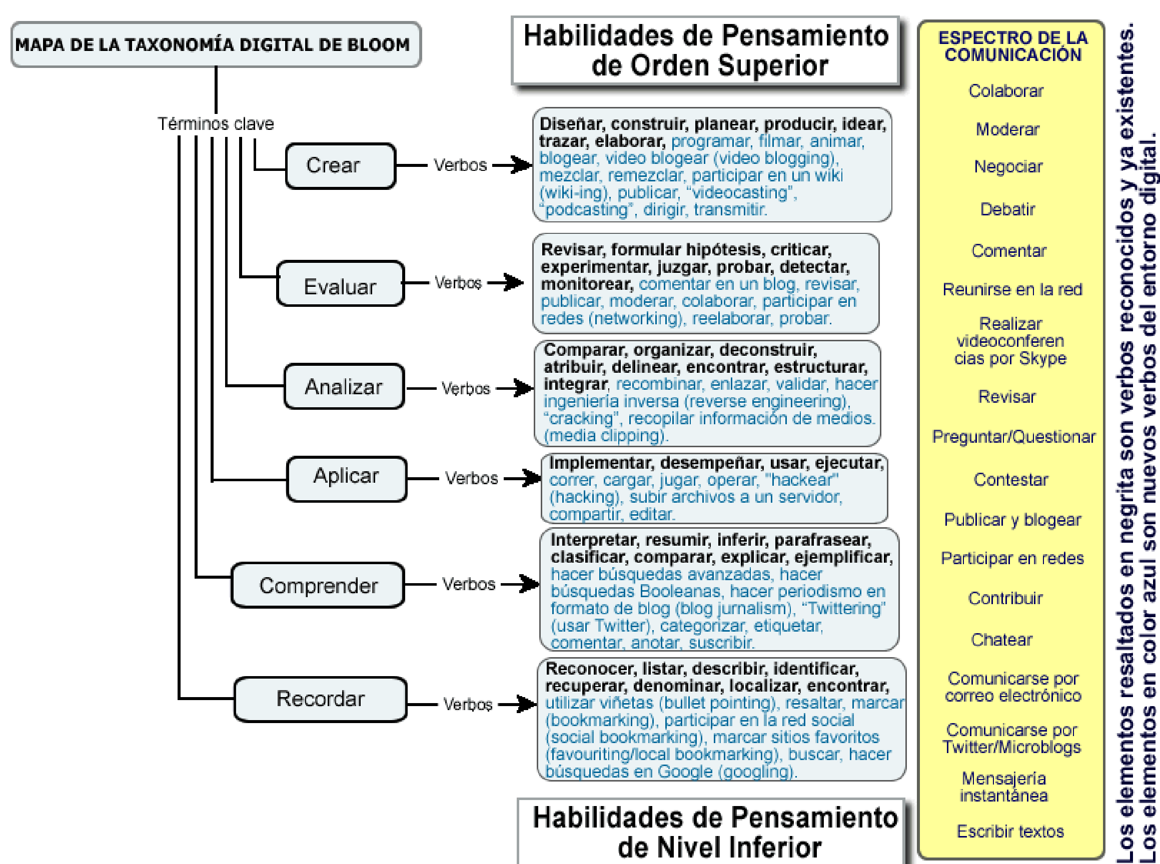


Figura 17. Taxonomía de Bloom para la Era Digital

De acuerdo con la Taxonomía de Bloom para la era digital[81], las habilidades de pensamiento siguen un orden jerárquico, de inferior a superior:

- **Recordar**– Reconocer, listar, describir, identificar, recuperar, denominar, localizar, encontrar.

---

[81] http://www.eduteka.org/TaxonomiaBloomDigital.php

- **Comprender**– Interpretar, resumir, parafrasear, inferir, clasificar, comparar, ejemplificar, explicar.
- **Aplicar**– Implementar, desempeñar, ejecutar, usar.
- **Analizar**– Comparar, delinear, atribuir, integrar, deconstruir, estructurar, organizar, encontrar.
- **Evaluar**– Revisar, formular hipótesis, criticar, experimentar, juzgar, probar, detectar, monitorear.
- **Crear**– Diseñar, construir, idear, trazar, planear, producir, elaborar.

3.4.2 Dominio Afectivo

Aprender a ser. Todos los seres humanos deben estar en condiciones, en particular gracias a la educación recibida en su juventud, de dotarse de un pensamiento autónomo, crítico y de elaborar un juicio propio, para determinar por sí mismos qué deben hacer en las diferentes circunstancias de la vida.

En este modelo se toman en cuenta diversas taxonomías de objetivos para concretizar los dominios de aprendizaje. Determina el modo como la gente reacciona emocionalmente, su habilidad para sentir el dolor o la alegría de otro ser viviente. Los objetivos afectivos apuntan típicamente a la conciencia y crecimiento en actitud, emoción y sentimientos.

Hay cinco niveles en el dominio afectivo. Yendo de los procesos de orden inferior a los superiores, son:

- **Recepción** - El nivel más bajo; el humano presta atención en forma pasiva. Sin este nivel no puede haber aprendizaje.
- **Respuesta** - El humano participa activamente en el proceso de aprendizaje, no sólo atiende a estímulos, el humano también reacciona de algún modo.
- **Valoración** - El humano asigna un valor a un objeto, fenómeno o información.
- **Organización** - Los humanos pueden agrupar diferentes valores, informaciones e ideas y acomodarlas dentro de su propio esquema; comparando, relacionando y elaborando lo que han aprendido.
- **Caracterización** - El humano cuenta con un valor particular o creencia que ahora ejerce influencia en su comportamiento de modo que se torna una característica.

3.4.3 Dominio Psicomotor

Aprender a hacer. Está estrechamente vinculado a la cuestión de la formación profesional. Aprender a hacer a fin de adquirir no sólo una calificación profesional sino, más generalmente, una competencia que capacite al individuo para hacer frente a gran número de situaciones y a trabajar en equipo. Pero, también aprender a hacer en el marco de las distintas experiencias sociales o de trabajo que se ofrecen a los jóvenes, ya sea espontáneamente a causa del contexto social o nacional; o, formalmente por el desarrollo de la enseñanza.

Los objetivos psicomotores generalmente se basan en el cambio desarrollado en la conducta o habilidades. Comprende los siguientes niveles:

- Percepción
- Disposición
- Mecanización
- Respuesta compleja
- Adaptación
- Creación

3.4.4 Dominio Relacional o Social

Hace referencia a aprender a vivir y aprender a vivir con los demás (convivir), este aprendizaje constituye una de las principales empresas de la educación contemporánea. Aprender a vivir en sociedad desarrollando la comprensión del otro y la percepción de las formas de interdependencia respetando a la naturaleza; y, a los valores humanos y éticos.

3.5 Principios Generales del Aprendizaje y la Motivación

El término motivación hace alusión al aspecto en virtud del cual una persona, como organismo vivo, es una realidad auto dinámica que le diferencia de los seres inertes. El organismo vivo se distingue de los que no lo son porque puede moverse a sí mismo. La motivación trata esos determinantes que hacen que la persona se comporte de una determinada manera teniendo en sí mismo el principio de su propio movimiento.

En el contexto cognitivo, la motivación, es el interés que tiene una persona por su propio aprendizaje o por las actividades que le conducen a él. El interés se puede adquirir, mantener o aumentar en función de elementos intrínsecos y extrínsecos. En esta perspectiva, los siguientes pueden considerarse como principios generales de motivación para el aprendizaje:

- **Significancia**: Una persona probablemente estará motivada a aprender aquello que le resulta significativo.
- **Prerrequisitos**: Es más probable que una persona aprenda algo si cumple con todos los prerrequisitos.
- **Modelamiento**: Es más probable que una persona adquiera una nueva conducta si se le presenta un modelo de desempeño que debe observar o imitar.
- **Comunicación Abierta**: Una persona tiene mayor probabilidad de aprender si la presentación está estructurada de tal forma que los mensajes del facilitador están abiertos a la inspección.
- **Innovación**: Una persona tiene mayor probabilidad de aprender si se atrae su atención mediante presentaciones relativamente novedosas.
- **Práctica Activa Adecuada**: Es más probable que una persona aprenda si toma parte activa en la práctica encaminada a alcanzar un objetivo de aprendizaje.

- **Práctica Distribuida**: Una persona tiene mayor probabilidad de aprender si su práctica es programada en períodos cortos distribuidos en el tiempo.
- **Desvanecimiento**: Una persona tiene mayor probabilidad de aprender si las indicaciones del aprendizaje son retiradas gradualmente.
- **Condiciones y Consecuencias Agradables**: Una persona tiene mayor probabilidad de continuar aprendiendo si las condiciones del aprendizaje son agradables.

3.5.1 La Pirámide del Aprendizaje

La pirámide del aprendizaje (*Learning Pyramid*), concepto desarrollado por el *Institute for Applied Behavioral Sciences*[82] plantea a los educadores la clave para promover aprendizajes efectivos.

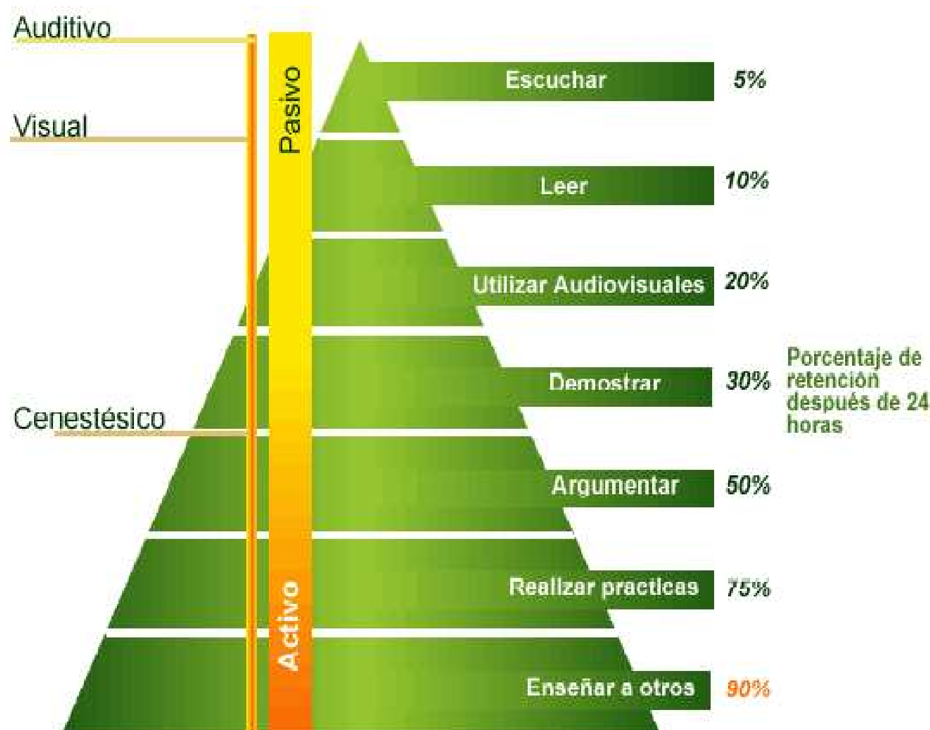


Figura 18. Pirámide del Aprendizaje

De la mencionada pirámide, se deduce que las clases expositivas o magistrales que utilizan una mayoría significativa de educadores sólo producen un 5 % de retención, esto porque el canal de transferencia es sólo auditivo, por tanto, el aprendizaje es pasivo; por el contrario, si se plantean experiencias a través de proyectos, problemas, casos, etc. que motiven a los alumnos a realizar actividades prácticas, ejercicios; entonces, se puede lograr un 75 % de retención, dado que el aprendizaje integra otros sentidos, por tanto, es activo[83].

En consecuencia, se entiende que estrategias didácticas tradicionales (exposición, diálogo, lectura, etc.), aún predominantes en un buen porcentaje de educadores son poco efectivas

---

[82] http://www.ntl.org/
[83] Blair, Cody. *Secrets Smart Students Know*. (eBook). http://studyprof.com

en el aprendizaje de los alumnos; por lo que se tienen que combinar con estrategias activas en la que los alumnos encuentren oportunidades para **aprender haciendo**. Lo propuesto no es nuevo, **John Dewey**[84] lo recomendó a inicios del Siglo XX, cuando afirmó que: "*la educación no es un asunto de narrar y escuchar, sino un proceso activo de construcción*"[85].

3.6 Tipos de Aprendizaje

El aprendizaje es un cambio en la conducta relativamente permanente, que ocurre como resultado de la experiencia personal, pero que se refuerza con la interacción social. Desde este punto de vista el aprendizaje es un proceso social. El sujeto aprende de los otros y con los otros. En esta interacción comparte aprendizajes y construye nuevos conocimientos..

3.6.1 Aprendizaje Memorístico

Surge cuando la tarea del aprendizaje consta de asociaciones puramente arbitrarias o cuando el sujeto lo hace arbitrariamente.

- Los hechos o datos se memorizan sin comprenderlos
- Se memorizan de forma repetitiva
- Si no se lleva a la práctica lo que se ha memorizado se olvida
- A mayor volumen de datos más difícil es la memorización de éstos
- El que los datos estén ordenados según algún criterio, ejemplo por bloques, facilita la memorización

3.6.2 Aprendizaje Receptivo

Una persona recibe el contenido que ha de internalizar, sobre todo por la explicación del profesor, el material impreso, la información audiovisual, los computadores.

- Es un aprendizaje por instrucción expositiva que comunica el contenido que va a ser aprendido en su forma final
- Se debe dar una instrucción que active en las personas los conocimientos previos necesarios, es decir, hacer un puente cognitivo entre los conocimientos previos y los nuevos
- Se debe hacer una presentación de los contenidos de manera estructurada y con una organización explícita que capte el interés de las personas

3.6.3 Aprendizaje por Descubrimiento

Una persona debe descubrir el material por sí mismo, antes de incorporarlo a su estructura cognitiva. Este aprendizaje por descubrimiento puede ser guiado o tutorado por el profesor.

---

[84] **http://en.wikipedia.org/wiki/John_Dewey**
[85] Dewey, John. *Democracy and Education: An Introduction to the Philosophy of Education*. Macmillan, USA, 1916.

- La persona construye sus conocimientos de una forma autónoma, sin la ayuda permanente del profesor
- Se exige mayor participación de la persona, ya que ellas son las que buscan
- Requiere un método de búsqueda activa por parte de la persona
- El profesor da las ideas principales, los objetivos, las metas
- El profesor es un mediador y guía y serán las personas quienes recorran el camino y alcancen los objetivos propuestos
- Es un aprendizaje útil, ya que cuando se lleva a cabo de modo eficaz, asegura un conocimiento significativo y fomenta hábitos de investigación y rigor en las personas

La principal desventaja es que emplea mucho tiempo, es por eso que no es un aprendizaje muy frecuente.

3.6.4 Aprendizaje Significativo

Se da cuando las tareas están interrelacionadas de manera congruente y el sujeto decide aprender así. En este caso la persona es el propio conductor de su conocimiento relacionado con los conceptos a aprender.

- Se aprenden conceptos. Existe una comprensión de lo que se aprende
- Como existe una comprensión de lo aprendido, es difícil que se olvide
- Los contenidos de cualquier asignatura deben poseer una organización conceptual interna, que mantengan coherencia todos los elementos entre sí
- La organización conceptual debe estar en un vocabulario que las personas lo entiendan
- El profesor debe conocer las ideas previas que las personas tienen sobre el tema a tratar

3.6.5 Aprendizaje por Modelado de Procedimientos

Consiste en conocer las formas de actuar, de usar conocimientos y de aplicar esas formas para conocer más. La idea es poner en práctica lo aprendido. Por ejemplo, no basta con saberse las fórmulas matemáticas si no se saben aplicar. Este aprendizaje quiere decir: "Primero lo haré yo (profesor), después lo haremos juntos, después lo harás tú solo"

- Se requiere que el profesor dé claras instrucciones y los beneficios de este aprendizaje.
- Los alumnos deben poner atención a todas las instrucciones dadas por el profesor.

El riesgo que se corre con este tipo de aprendizaje es que el alumno aprenda de memoria o por imitación

3.6.6 Nuevas Formas de Aprendizaje

El conductismo, el cognitivismo y el constructivismo son las tres grandes teorías de aprendizaje utilizadas más a menudo en la creación de ambientes instruccionales. Estas

teorías, sin embargo, fueron desarrolladas en una época en la que el aprendizaje no había sido impactado por la tecnología. No hacen referencia al aprendizaje que ocurre por fuera de las personas (aprendizaje que es almacenado y manipulado por la tecnología). También fallan al describir cómo ocurre el aprendizaje al interior de las organizaciones. En los últimos veinte años, la tecnología ha reorganizado la forma en la que se vive, comunica y se aprende.

El **Conectivismo de Siemens**[86] integra la tecnología y la identificación de conexiones como actividades de aprendizaje, y emerge como una teoría de aprendizaje más apropiada para la edad digital. Plantea que no es posible experimentar y adquirir personalmente el aprendizaje que necesitamos para actuar, sino que derivamos nuestra competencia de la formación de conexiones. El **conectivismo** presenta un modelo de aprendizaje que reconoce la dinámica aleatoria, difusa, caótica y compleja propia de la sociedad en la era digital, en donde el aprendizaje ha dejado de ser una actividad interna e individual. La forma en la cual trabajan y funcionan las personas se fortalece cuando se usan herramientas de TI. Los principios del conectivismo son:

- El aprendizaje y el conocimiento dependen de la diversidad de opiniones.
- El aprendizaje es un proceso de conectar nodos o fuentes de información especializados.
- El aprendizaje puede residir en dispositivos no humanos.
- La capacidad de saber más es más crítica que aquello que se sabe en un momento dado.
- La alimentación y mantenimiento de las conexiones es necesaria para facilitar el aprendizaje continuo.
- La habilidad de ver conexiones entre áreas, ideas y conceptos es una habilidad clave.
- La actualización (conocimiento preciso y actual) es la intención de todas las actividades conectivistas de aprendizaje.
- La toma de decisiones es, en sí misma, un proceso de aprendizaje. El acto de escoger qué aprender y el significado de la información que se recibe, es visto a través del lente de una realidad cambiante. Una decisión correcta hoy, puede estar equivocada mañana debido a alteraciones en el entorno informativo que afecta la decisión.

E-Learning, es un término dado a una forma de aprendizaje asistido por tecnologías de la información. Este tipo de aprendizaje presenta las siguientes ventajas:

- Fomenta el conectivismo y el uso eficiente de las TI.
- Desarrolla la creatividad y la motivación de los estudiantes.

[86] http://humanismoyconectividad.wordpress.com/2009/01/14/conectivismo-siemens/

- Permite la adopción de conceptos y el descubrimiento de herramientas de aprendizaje.
- Facilita el intercambio de opiniones y experiencias a través de las TI (Aprendizaje Social).

3.7 Estilos de Aprendizaje

Los Estilos de Aprendizaje son los rasgos cognitivos, afectivos y fisiológicos, que sirven como indicadores relativamente estables, de cómo las personas perciben, interaccionan y responden a sus ambientes de aprendizaje[87].

Cuando se habla de estilos de aprendizaje se tienen en cuenta los rasgos cognitivos, se incluyen los estudios de psicología cognitiva que explica la diferencia en las personas respecto a la forma de conocer. Este aspecto cognitivo es el que caracteriza y se expresa en los estilos cognitivos.

El estudio sobre los Estilos de Aprendizaje se enmarca dentro de los enfoques pedagógicos contemporáneos que insisten en la creatividad, **aprender a aprender**: Desarrollar el conocimiento y destreza necesarios para aprender con efectividad en cualquier situación en que uno se encuentre. Una persona ha aprendido a aprender si sabe:

- Cómo controlar el propio aprendizaje.
- Cómo desarrollar un plan personal de aprendizaje.
- Cómo diagnosticar sus puntos fuertes y débiles como estudiante.
- Cómo describir su Estilo de Aprendizaje.
- En qué condiciones aprende mejor.
- Cómo aprender de la experiencia de cada día.
- Cómo aprender de la radio, TV, prensa, Internet.
- Cómo participar en grupos de discusión y de resolución de problemas.
- Cómo aprovechar al máximo una conferencia o un curso.
- Cómo aprender de un tutor.
- Cómo usar la intuición para el aprendizaje.

Los componentes en la idea de Aprender a Aprender, son:

- Necesidades de una persona (lo que necesita conocer y ser capaz de hacer para tener éxito en el aprendizaje).
- Estilos de Aprendizaje (preferencias y tendencias altamente individualizadas de una persona que influye en su aprendizaje).

---

[87] Keefe, James W. *Profiling and utilizing learning style*. National Association of Secondary School Principals, Virginia. USA, 1988.

- Formación (actividad organizada para aumentar la competencia de una persona en el aprendizaje).
    - Comprensión general que facilite una base de actitud positiva y motivación como necesita el aprendizaje.
    - Destrezas básicas: leer, escribir, matemáticas y, en nuestro tiempo, además, saber escuchar y alfabetización informática.
    - Autoconocimiento: Puntos fuertes y puntos débiles de uno mismo, preferencias personales por los métodos, estructuras y ambientes de aprendizaje.
    - Procesos educativos para tres modos de aprendizaje: auto dirigido, en grupo o institucional.
    - Interrelaciones entre Componentes.

Peter Honey y **Allan Mumford** en 1986[88], crearon un instrumento para evaluar estilos de aprendizaje. Para generar el instrumento parten adecuando la teoría y los cuestionarios de David Kolb[89], tratando de buscar una herramienta más completa que oriente hacia la mejora del aprendizaje. Lo ideal, debería ser que todo el mundo fuera capaz, de igual manera, de experimentar, reflexionar, elaborar hipótesis y aplicar. Pero lo cierto es que cada persona es más capaz de una cosa que de otra.

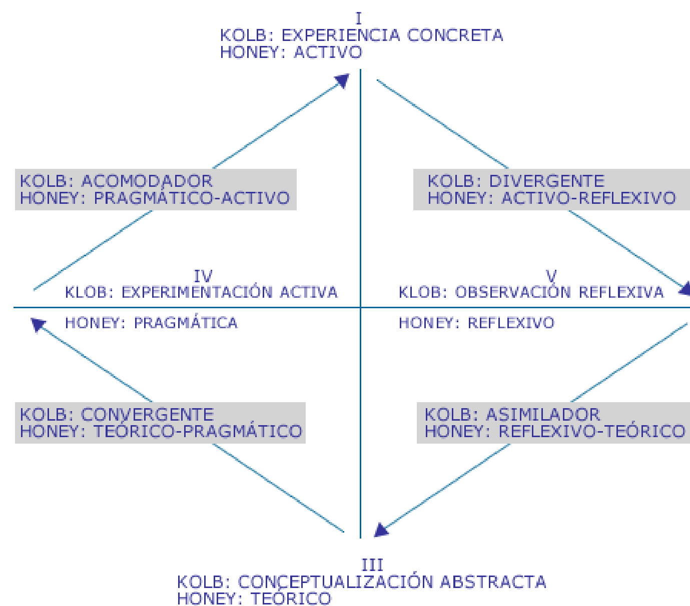


Figura 19. Estilos de Aprendizaje

[88] Honey, Peter, y Munford, Alan: *Using your learning styles*. Maidenhead. **www.peterhoney.com**. 1986
[89] Kolb. D. A. and Fry, R. (1975) *Toward an applied theory of experiential learning*. in C. Cooper (ed.) Theories of Group Process, London: John Wiley

Los estilos propuestos por Honey y Mumford son cuatro: activo, reflexivo, teórico y pragmático. A partir de esta descripción de estilos, Alonso, Gallego y Honey[90] publican una lista de características que determina con claridad el campo de destrezas de cada estilo, y desarrollan el denominado Cuestionario Honey – Alonso de Estilos de Aprendizaje (**CHAEA**)[91].

3.8 Teoría de Aprendizaje de Robert Gagné

Robert Gagné[92], psicólogo y pedagogo norteamericano, propone una teoría de aprendizaje, basada en un modelo de procesamiento de información, con un enfoque sistémico.[93,94,95,96] Fusiona el conductismo, el cognitivismo, el aprendizaje significativo y el aprendizaje social. Su teoría de **condiciones de aprendizaje** consiste de tres componentes:

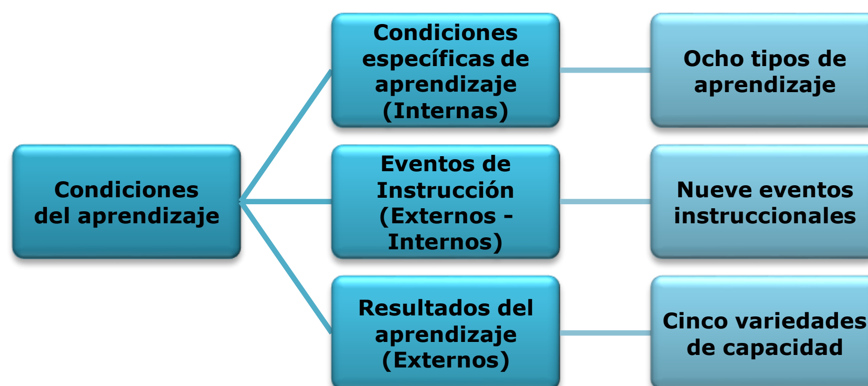


Figura 20. Teoría del aprendizaje según Robert Gagné

1. **Condiciones específicas de aprendizaje**.- Tipos de aprendizaje jerarquizados en función de su grado de ascendente de complejidad. Los de orden superior se construyen en base a los de orden inferior. Los cuatro primeros, que son de orden más bajo, se enfocan más en aspectos conductuales, mientras que los siguientes cuatro de orden superior, se enfocan en aspectos cognitivos.
    - **Aprendizaje de señales**. Puede ser equivalente al condicionamiento clásico o de reflejos. Aprender a responder a una señal. Usualmente la respuesta es emotiva.
    - **Aprendizaje de estímulo-respuesta (E/R)**. Aproximadamente equivalente al condicionamiento instrumental u operante. Dar respuestas precisas a estímulos o señales determinadas.

[90] Alonso, C.M; Gallego, D.J.; Honey, P: *Los estilos de aprendizaje. Procedimientos de Diagnóstico y Mejora*. 4ª Edición. Ediciones Mensajero, Bilbao, España, 1999.
[91] http://www.estilosdeaprendizaje.es/chaea/chaea.htm
[92] http://es.wikipedia.org/wiki/Robert_M._Gagn%C3%A9
[93] Gagné, R. M. *The conditions of learning* (1st ed.). New York: Holt, Rinehart, & Winston, 1966.
[94] Gagné, R. M. *Learning Hierarchies*. Educational Psychologist. American Psychological Association, 1968.
[95] Gagné, R. M. *Domains of Learning*. Interchange. 3, 1972, pp 1-8.
[96] Gagné, R.M., & White, R.T. *Memory Structures and learning outcomes*. Review of Educational Research, 48, 1978. pp. 187-222.

- **Aprendizaje de encadenamiento psicomotor**. Ocurre cuando se forma una cadena de estímulos – respuestas. Se aprende a seguir procedimientos. Capacidad para concatenar 2 o más conjuntos estímulo – respuesta.
- **Aprendizaje de asociación verbal (E/R: en el área verbal)**. Uso de terminología en encadenamientos verbales.
- **Aprendizaje de discriminación múltiple**. Saber distinguir entre estímulos similares. Generar respuestas diferentes para cada tipo de estímulo, aun cuando parezcan perceptualmente similares.
- **Aprendizaje de conceptos**. Respuesta única, común, a una clase completa de estímulos.
- **Aprendizaje de principios**. Encadenamiento de conceptos, aplicación de reglas.
- **Resolución de problemas**. Todos los tipos anteriores deben haberse completado, antes de que este ocurra.

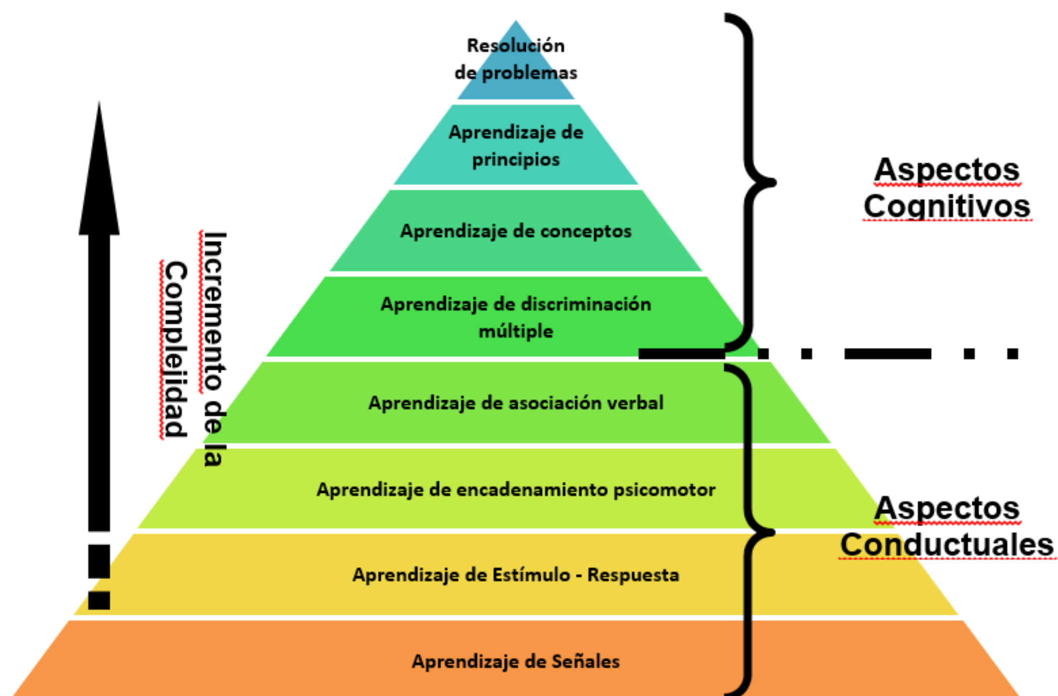


Figura 21. Jerarquía de los tipos de aprendizaje, según Robert Gagné

2. **Eventos de Instrucción**.- Cada instrucción consiste de un grupo de eventos externos al estudiante, diseñados para estimular sus procesos internos de aprendizaje. La teoría de condiciones de aprendizaje, propone 9 eventos instruccionales, con sus respectivas actividades:
   - Captar la atención.
   - Informar los objetivos.
   - Estimular el conocimiento previo.
   - Presentar material nuevo.
   - Guiar el aprendizaje.

- Suscitar el rendimiento individual.
- Proporcionar retroalimentación.
- Evaluar la eficacia del rendimiento.
- Incrementar la retención.

3. **Taxonomía de resultados del aprendizaje (Dominios del Aprendizaje)**.- Basado en los 8 tipos de aprendizaje y en los 9 eventos de instrucción, **Robert Gagné** señala 5 variedades de capacidades que pueden ser aprendidas:
   - **Destrezas motoras**. Estas capacidades son muy importantes en ciertas áreas del aprendizaje, en las cuales se requiere uniformidad y regularidad en las respuestas
   - **Información verbal**. A la cual estamos expuestos desde nuestro nacimiento; además debemos demostrar una conducta después que recibimos esta información (hacer oraciones, frases, etc.) Su recuperación es facilitada generalmente por sugerencias externas. Lo más destacable del aprendizaje de esta información es que posee un amplio contexto significativo, mediante el cual la podemos asociar a información ya existente.
   - **Destrezas intelectuales**. Comienza al adquirir discriminaciones y cadenas simples, hasta llegar a conceptos y reglas. Podemos hacer cosas con los símbolos y comenzar a entender qué hacer con la información. En este aprendizaje necesitamos combinar destreza intelectual e información verbal previamente aprendida.
   - **Actitudes**. Estas son las capacidades que influyen sobre las acciones individuales de las personas. Es difícil enseñar actitudes, y la mayoría de ellas debe ser adquirida y reforzada en la educación formal. Es necesario estudiar las actitudes negativas y las positivas, campo que fue llamado por Bloom como dominio afectivo. Gagné define las actitudes como un estado interno, pero medible sólo a través de la conducta manifiesta.
   - **Estrategias cognoscitivas**. Son destrezas de organización interna, que rigen el comportamiento del individuo con relación a su atención, lectura, memoria, pensamiento, etc.

3.9 Aprendizaje en Sistemas Computarizados

El modelo de **Robert Gagné**, ha sido adaptado para diseñar procesos de aprendizaje de un sistema computarizado. Desde la perspectiva abstracta de un sistema, el aprendizaje, se considera como un proceso de información adaptable que introduce cambios en el mismo de tal manera que la próxima vez pueda realizar la misma tarea o tareas de similar tipo de un modo más eficiente y eficaz.

Para adaptarse, el sistema debe interactuar con el entorno en el que está embebido, aprender un modelo del mismo y mantenerlo suficientemente consistente con el mundo real como para satisfacer los objetivos de evolución y mejoramiento continuo. El entorno está representado por:

- El estado conocido o conocimiento *a priori*.
- Las observaciones o mediciones de las que se seleccionan muestras para adaptar el sistema.

El aprendizaje en sistemas, es similar a un proceso de resolución de problemas y cubre un amplio espectro: desde perfeccionamiento de la habilidad hasta adquisición de conocimiento. El proceso por el cual el aprendizaje cambia la conducta de un sistema, pueden ser sintetizado como:

- Los estímulos adquieren nuevos significados.
- Se aprenden nuevas respuestas ante los estímulos.
- Se elimina lo que no es esencial y la conducta se torna mejor integrada y más autónoma.

4 CAPÍTULO IV: MODELOS DE INTELIGENCIA ARTIFICIAL

4.1 Introducción

Históricamente, el ser humano se utilizó a sí mismo como el modelo *de facto* en sus variados intentos de desarrollar máquinas inteligentes. De hecho, la era de la inteligencia artificial se inició bajo el supuesto que la inteligencia humana podía ser descrita con suficiente precisión como para ser simulada en una máquina[97].

Pamela McCorduck, en 1979, publica *Machines Who Think*[98]. Esta obra, concebida como una historia de la inteligencia artificial, cubre desde los sueños de los poetas clásicos griegos y profetas hebreos, hasta su realización como una ciencia, en el Siglo XX. Delinea los primeros intentos de filósofos y, posteriormente, de psicólogos de definir la mente, mientras paralelamente matemáticos intentaban desarrollar modelos rigurosos para la lógica humana. Demuestra que el modelo dominante de estos pensadores fue, desde la perspectiva de la física, **energía en un sistema cerrado**. Y, no fue sino hasta el aparecimiento de la **cibernética**, con la que se introdujo un nuevo modelo, **información en un sistema abierto**, que se propició el uso del computador como un medio exitoso para desarrollar comportamiento inteligente.

La creación de algunas formas de inteligencia artificial basadas en computador, que pueden razonar y resolver problemas sólo en un limitado dominio, efectivamente, actúan como si fueran inteligentes, pero es evidente que no poseen verdadera inteligencia. A esta **inteligencia de propósito específico**, se definió como **inteligencia artificial débil**[99].

La mayor parte de la investigación realizada durante la segunda mitad del Siglo XX, cae dentro de esta definición y constituye la denominada **inteligencia artificial clásica**, que propuso modelos de **computación simbólica**, basados en el pensamiento humano y su estilo heurístico para resolver problemas y tomar decisiones, las ciencias cognitivas, la cibernética y la lógica.

4.2 Modelos de Inteligencia Artificial Clásica

La inteligencia artificial clásica, comprende un conjunto de modelos de sistemas inteligentes, caracterizados por:

---

[97] McCarthy, John; Minsky, Marvin; Rochester, Nathan; Shannon, Claude. **A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence**. 1955

[98] **http://www.pamelamc.com/html/machines_who_think.html**

[99] **http://en.wikipedia.org/wiki/Weak_AI**

a) La representación formal del problema a resolver, como una red semántica; y,
b) Su capacidad de procesamiento simbólico, basada en algoritmos de búsqueda de soluciones.

De aquí se deriva el nombre de **computación simbólica** para la implementación tecnológica de las aplicaciones basadas en los modelos clásicos de inteligencia artificial.

La **Computación Simbólica** se define como un área de investigación de carácter interdisciplinar, que se enmarca en el ámbito común de actuación de campos como la Matemática y las Ciencias de la Computación, y cuyo cometido principal es el desarrollo, construcción y análisis de algoritmos efectivos para la manipulación de objetos simbólicos susceptibles de ser representados en un ordenador, con énfasis en los cálculos correspondientes a objetos de entidad matemática y enfocado a sus aplicaciones, no solo en la propia Matemática, sino también en otras ramas de la Ciencia (como la Física, la Química, la Biología, etc.) e incluso en la Industria (redes eléctricas, modelado de automóviles, redes de transporte, tolerancia geométrica, diseño geométrico asistido por ordenador, robótica, etc.).

En esencia, la computación simbólica comprende la solución algorítmica de problemas relacionados con **objetos simbólicos**. Se consideran objetos simbólicos, a todos los objetos matemáticos y sus representaciones computacionales; por ejemplo aquellos que corresponden al algebra computacional y a la lógica computacional.

En la computación simbólica un problema puede definirse como la búsqueda de una situación objetivo en un entorno determinado, con su correspondiente conjunto de medios que permitirían alcanzarla a partir de una situación dada. Para la formulación de un problema se requieren acciones de observación, descripción, explicación y predicción. Esto facilita y permite ejecutar con éxito los pasos del siguiente proceso iterativo:

- Identificación del problema
- Percepción de la situación que rodea al problema identificado
- Análisis y definición del problema
- Definición de los objetivos a ser alcanzados por la solución
- Búsqueda de alternativas de solución o cursos de acción
- Comparación y evaluación de las alternativas identificadas
- Selección de la alternativa más adecuada para el logro de los objetivos
- Implementación de la alternativa seleccionada
- Pruebas Experimentales
- Análisis de Resultados y Conclusiones

La ejecución del proceso iterativo que conlleva a obtener una representación formal de un problema en computación simbólica, requiere de:

a) Identificación de fuentes para adquisición del conocimiento necesario (Documentos, experiencias, procesos, expertos humanos, etc.)
b) Adquisición y clasificación del conocimiento (Para simplificar el proceso de búsqueda de las soluciones y para resolver el problema).
c) Representación simbólica del conocimiento (Proceso de extraer, transformar, traducir y transferir la experticia para resolver problemas desde una fuente de conocimiento hacia una forma procesable por un programa de computadora).

En general, una representación se puede definir como un conjunto de convenciones sobre la forma de describir alguna cosa. Una representación tiene 4 elementos importantes:

- **Léxico**.- Símbolos permitidos en el vocabulario.
- **Estructura**.- Restricciones para el ordenamiento de los símbolos.
- **Operadores**.- Procedimientos para crear, modificar y utilizar descripciones.
- **Semántica**.- Forma de asociar el significado con las descripciones.

Una de las representaciones más utilizadas para problemas de inteligencia artificial son las redes semánticas. Una red semántica es una notación gráfica para representar conocimiento en patrones caracterizados por:

- Nodos, enlaces y etiquetas (**Léxico**)
- Nodos conectados por enlaces etiquetados (**Estructura**)
- Constructores, lectores, escritores y destructores (**Operadores**)
- Significado de nodos y enlaces (**Semántica**)

Los principales tipos de redes semánticas, son[100]:

- **Redes de Definición**, destacan la relación de subtipo o es-una entre un tipo de concepto y un subtipo recién definido. La red resultante, llamada también una generalización, cumple la regla de herencia para propagar las propiedades definidas para un supertipo a todos sus subtipos. Dado que las definiciones son verdaderas por definición, la información en estas redes a menudo se asume que es necesariamente verdadera.
- **Redes Asertivas**, están diseñadas para afirmar proposiciones. A diferencia de redes de definición, la información en una red de aserciones se supone contingentemente verdadera, a menos que explícitamente se marque con un operador modal. Se han propuesto algunas redes asertivas como modelos de las estructuras conceptuales subyacentes en la semántica de lenguaje natural.
- **Redes de Implicación**, utilizan implicación como la relación principal para conectar nodos. Se puede utilizar para representar patrones de creencia, causalidad o inferencia.

[100] John F. Sowa. *Semantic Networks*. http://www.jfsowa.com/pubs/semnet.htm

- **Redes Ejecutables**, incluyen algún mecanismo, como pasar marcadores o procedimientos adjuntos, que pueden realizar inferencias, propagar mensajes, o buscar patrones y asociaciones.
- **Redes de Aprendizaje**, construyen o amplían sus representaciones con la adquisición de conocimiento a través de ejemplos. El nuevo conocimiento puede cambiar la antigua red mediante la adición y eliminación de nodos y arcos o de valores numéricos, llamados pesos, asociados con los nodos y arcos.
- **Redes Híbridas**, combinan dos o más de las técnicas anteriores, en una única red o en redes separadas, pero que estrechamente interactúan entre sí.

La representación de un problema como una red semántica, en computación simbólica, permite definir formalmente a un problema y resolverlo utilizando un conjunto de operadores y un proceso de búsqueda. La representación puede ser:

- **Explícita**.- Descripción completa, transparente y concisa de una red semántica. Agrupa todos los objetos y sus relaciones, poniendo de manifiesto las restricciones inherentes del problema.
- **Implícita**.- Descripción de una red semántica compuesta por un estado inicial y un mecanismo para generar estados sucesores.

Principio de Representación

Una vez que un problema es descrito por una buena representación, el problema está casi resuelto.

En esencia, la definición formal de problemas en computación simbólica, implica:

- Escoger una red semántica apropiada para modelar el tipo de problema identificado.
- Identificar y seleccionar los posibles estados iniciales.
- Especificar un conjunto de acciones para alcanzar o generar nuevos estados (operadores).
- Identificar la meta o metas deseadas (estados objetivos).
- Establecer una función de costo de ruta desde el estado inicial al estado objetivo.
- Definir criterios de aceptabilidad de las metas alcanzadas (validación de meta) o de las trayectorias encontradas (validación de trayectoria), como soluciones.
- Seleccionar y aplicar de la mejor técnica de solución. Una técnica de inteligencia artificial, es un método que utiliza conocimiento, abstracción y procesos de búsqueda para resolver problemas complejos. La selección y aplicación de la mejor técnica depende de la naturaleza del problema y de la representación adoptada para el mismo.
- Evaluar los resultados obtenidos. ¿La meta se alcanzó de forma definitiva y clara? ¿El procedimiento aplicado es confiable? ¿Las regularidades o restricciones son claramente

identificables? ¿El problema resuelto es un problema real o una simplificación del mismo? ¿Los resultados son generalizables para la solución de otros problemas?

4.2.1 Simulación Cognitiva

Herbert Simon y **Allen Newell** estudiaron las habilidades de los humanos para resolver problemas y trataron de formalizarlas simbólicamente.

Los resultados de estos estudios conformaron la base para el desarrollo de las ciencias cognitivas, investigación de operaciones, las ciencias administrativas y de sistemas inteligentes basados en la simulación cognitiva humana (**Computación Heurística**)[101].

El modelo del procesador cognitivo de Newell y Simon considera la mente como un sistema de procesamiento de la información.

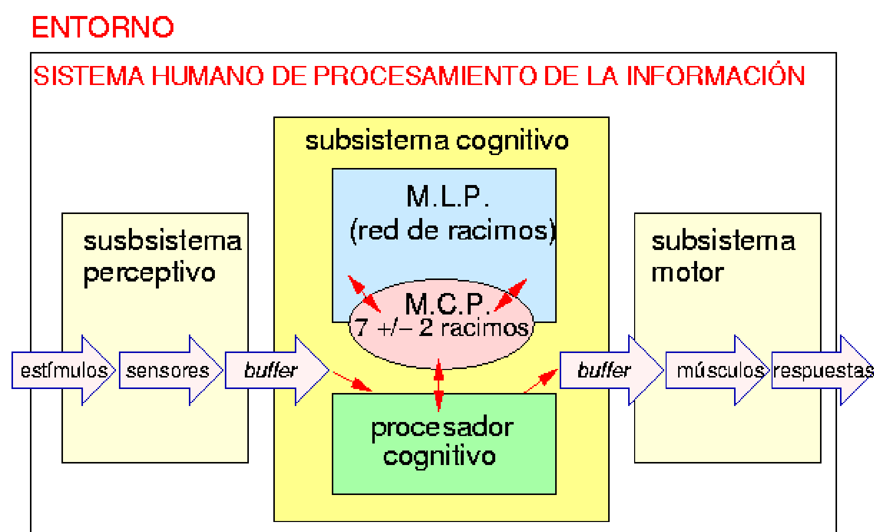


Figura 22. Modelo de procesamiento de la información

La analogía con la estructura de un ordenador es evidente. Basándose en datos experimentales, el modelo incluye la hipótesis de que el procesador cognitivo selecciona preceptos y reacciona ante ellos en ciclos reconocimiento y acción de unos 70 milisegundos. Las transferencias entre la M.C.P (memoria a corto plazo) y la M.L.P (memoria a largo plazo) tardarían unos siete segundos. Estos números no son caprichosos. Están apoyados, como el número mágico 7±2, en diversos experimentos[102].

4.2.2 Sistemas Basados en Lógica

La Lógica Computacional juega un papel importante en muchas áreas de la informática, incluida la verificación de hardware y software, lenguajes de programación, bases de datos e Inteligencia Artificial.

---

[101] Newell, A., y Simon, H. A.: *Human problem solving*. Prentice-Hall, Englewood Cliffs, NJ., 1972.
[102] Miller, G A. *The Magical Number Seven, Plus or Minus Two Some Limits on Our Capacity for Processing Information*. Psychological Review © by the American Psychological Association. Vol. 101, No. 2, 1956, pp. 343-352

En el caso de la Inteligencia Artificial, la lógica es el fundamento de todos los métodos de representación del conocimiento y del razonamiento, especialmente en sistemas expertos, razonamiento con incertidumbre (encadenamiento de reglas, lógica difusa, etc.), procesado del lenguaje natural, razonamiento espacial y temporal, visión artificial, robótica, lógica epistémica, etc.

Los métodos de la lógica, especialmente de la lógica de predicados y de la lógica modal son los que más se utilizan hoy en día en ciencias de la computación, inteligencia artificial e ingeniería del software.

John McCarthy trató de representar la esencia del razonamiento abstracto y de la resolución de problemas, independientemente de si los humanos usamos o no estos algoritmos. Sus investigaciones se enfocaron en el uso de lógica formal para resolver la representación simbólica del conocimiento, el planeamiento y el aprendizaje. Esto dio lugar al desarrollo del lenguaje **PROLOG** y a la programación lógica.

4.2.3 Sistemas Basados en Conocimiento

Los Sistemas Basados en Conocimiento son aplicaciones relacionadas con la captura, codificación y utilización del conocimiento de expertos humanos. Los sistemas basados en conocimiento requieren de una interfaz, base de conocimientos y un motor de inferencia.

Para representar el conocimiento, utilizan comúnmente reglas **si-entonces**, pero también es posible usar otras representaciones del conocimiento como por ejemplo las redes semánticas.

Cuando las fuentes de conocimiento son muy complejas, el uso de métodos de aprendizaje automático puede ayudar a capturar y codificar el conocimiento necesario.

El Sistema que trata de alcanzar inteligencia a través de procesos que:

- Adquieren, representan, almacenan y recuperan conocimiento representado simbólicamente en un formato computacional;
- Procesan el conocimiento simbólico, con la ayuda de un motor de inferencia; y,
- Ejecutan acciones basadas en el proceso del conocimiento simbólico almacenado en su base de conocimientos.

Los principales representantes de este grupo son los denominados sistemas expertos.

4.3 Computación Sub Simbólica

En la computación sub-simbólica se trata de simular los elementos de más bajo nivel dentro de los procesos cognitivos, a fin de evidenciar el comportamiento inteligente.

Estos sistemas trabajan bajo conceptos fuertemente relacionados como la autonomía, el aprendizaje y la adaptación.

El enfoque sub-simbólico de la Inteligencia Artificial se caracteriza por crear sistemas con capacidad de aprendizaje. Estos sistemas incluyen procesos de desarrollo o aprendizaje iterativo, basado en datos experimentales. Los más conocidos en este grupo, son:

- **Los sistemas inteligentes computacionales**:
  - Computación neuronal
  - Computación evolutiva
- **Los sistemas basados en comportamiento**
  - Robótica
  - Agentes de software
  - Vida Artificial

4.4 La Nueva Inteligencia Artificial

En la última década del Siglo XX comenzaron a aparecer nuevas propuestas para la comprensión y la ingeniería de sistemas inteligentes. Nuevos campos de conocimiento tales como ciencia cognitiva embebida, ingeniería neuromórfica, vida artificial, robótica basada en comportamiento, robótica evolutiva e inteligencia de enjambre, cuestionan la validez de los modelos y métodos de la inteligencia artificial clásica que tratan de aproximar las características operativas y el desempeño de la inteligencia biológica.

Las teorías y métodos desarrollados por la **inteligencia artificial bio-inspirada**[103], consideran sistemas biológicos y artificiales que operan a diferentes escalas temporales y espaciales:

- En la **escala temporal**, los sistemas van desde los sistemas evolutivos que se desarrollan y evolucionan durante todo su ciclo de vida, hasta sistemas que interactúan en tiempo real con su entorno y con otros individuos:
  - Sistemas evolutivos
  - Sistemas que se desarrollan
  - Sistemas basados en comportamiento
- En la **escala espacial**, los sistemas van desde células, neuronas, organismos multicelulares, hasta sociedades de individuos
  - Sistemas celulares
  - Sistemas neuronales
  - Sistemas inmunológicos
  - Sistemas colectivos

---

[103] Floreano, D and Mattiussi, C. Bio-Inspired Artificial Intelligence. Theories, Methods and Technologies. The MIT Press, USA, 2008.

4.5 Agentes Inteligentes

Son sistemas que perciben su entorno, razonan y toman acciones de tal forma que maximizan sus oportunidades de éxito. Este paradigma permite que los investigadores estudien problemas complejos y busquen soluciones que son al mismo tiempo útiles y verificables.

La tecnología de los agentes inteligentes conforma la base de una nueva generación de sistemas computacionales. Las aplicaciones incluyen sistemas desarrollados para búsqueda de información masivamente distribuida en Internet, sistemas de información móviles, sistemas de *workflow* inteligentes, e infraestructura de información de apoyo a las operaciones y decisiones corporativas.

Se puede utilizar cualquier aproximación que funcione, esto es, pueden ser soluciones simbólicas, sub simbólicas, híbridas y otras nuevas que faciliten la solución del problema.

Parte fundamental en la nueva inteligencia artificial es el desarrollo de agentes inteligentes, integrando lo mejor de las teorías, modelos y tecnología clásicas y nuevas, e incluyendo la **lógica difusa** en los motores de inferencia y razonamiento.

5 CAPÍTULO V: SIMULACIÓN COGNITIVA

5.1 Introducción

El modelo clásico de inteligencia constituye la base para el desarrollo de sistemas inteligentes basados en la simulación cognitiva. En una publicación del año 1976[104], **Newell** y **Simon**, propusieron que la actividad inteligente, ya sea humana o artificial, es alcanzada mediante el uso de:

a) Un conjunto de símbolos para representar aspectos significativos del dominio del problema.
b) Operaciones sobre este conjunto de símbolos para generar potenciales soluciones para los problemas.
c) Búsqueda para seleccionar una solución de entre las potenciales soluciones.

5.2 El Sistema de Símbolos Físicos

Newell y Simon explican el concepto de **Sistema de Símbolos Físicos** y lo definen como:

> ***Un conjunto de entidades llamadas símbolos, que son patrones físicos que pueden funcionar como componentes de otro tipo de entidad llamada expresión (o estructura de símbolos). Una estructura de símbolos está formada por un número de instancias (señales o tokens) de símbolos relacionados de alguna forma física. En algún instante el sistema contendrá una colección de esas estructuras de símbolos.***

El sistema contiene también una colección de procesos que operan sobre expresiones para producir otras expresiones: procesos de creación, modificación, reproducción y destrucción. Un sistema de símbolos físicos es una máquina que produce a lo largo del tiempo una colección evolutiva de estructuras de símbolos. Este sistema existe en un mundo de objetos tan extenso como sus propias expresiones simbólicas. **La hipótesis de sistema de símbolos físicos** establece que:

> ***Un sistema de símbolos físicos posee los medios necesarios y suficientes para realizar una acción inteligente y genérica.***

---

[104] Newell A and Simon H. *Computer Science as Empirical Inquiry: Symbols and Search*. Communications of the ACM, March 1976, Vol 19, N° 3, pp. 113-126.

La hipótesis del Sistema de Símbolos Físicos representa una importante teoría clásica acerca de la inteligencia. Sirve de base para sustentar la creencia de que es posible crear sistemas computacionales inteligentes.

Sin embargo procesos sub-simbólicos como los que tienen lugar en las redes neuronales han cuestionado lo simbólico como tareas de bajo nivel. Lo que da lugar a pensar que los sistemas de símbolos físicos sólo sean capaces de modelar algunos aspectos de la inteligencia humana y no otros.

Otro paso importante relacionado con el sistema de símbolos físicos fue el desarrollo de estructuras y procesamiento computacional de listas. Estas estructuras contienen símbolos y listas que, a su vez, referencian a través de direcciones a otros símbolos o listas. El procesamiento de listas, dentro de las ciencias de computación, representa:

a) La creación de una memoria dinámica, añadiendo al conjunto de operaciones que cambiaban y remplazaban contenido, las de construcción y modificación de estructuras.
b) La abstracción de un computador como un conjunto de tipos de datos y operaciones apropiadas a ellos, independientemente de la máquina utilizada.
c) Un modelo de manipulación simbólica.

5.3 LISP

John McCarthy, en 1960, crea un sistema de programación denominado **LISP** (**LIS**t **P**rocessor)[105]. Fue diseñado para facilitar la experimentación con el sistema denominado ***Advice Taker***[106], en este sistema, la máquina puede ser instruida para manipular sentencias declarativas e imperativas y mostrar sentido común al ejecutar sus instrucciones.

La manipulación simbólica es el bloque constructivo básico de los programas de inteligencia artificial bajo el modelo clásico de inteligencia. Programas para manipulación simbólica pueden reconocer expresiones simbólicas particulares y pueden dividir expresiones para construir nuevas.

LISP es un lenguaje diseñado para manipulación simbólica, en contraste con los lenguajes de programación convencionales que están primordialmente diseñados para procesamiento numérico. En lugar de manipular estructuras de datos numéricos (como números y arreglos), los programas en LISP típicamente manipulan estructuras de datos simbólicos (como palabras y oraciones)[107].

---

[105] McCarthy, J. ***Recursive functions of symbolic expressions and their computation by machine, Part I***. Communications of the ACM Vol 3, N° 4, April 1960, pp. 184-195

[106] McCarthy, J. ***Programs with common sense***. Symposium on Mechanization of Thought Processes. National Physical Laboratory, Teddington, England, 1958.

[107] Banda, Hugo. *Programación en LISP – Curso Básico*. Departamento de Informática y Ciencias de Computación, Facultad de Ingeniería de Sistemas, Escuela Politécnica Nacional, Quito, 2003.

5.3.1 Expresiones Simbólicas: Átomos

Los elementos sintácticos del lenguaje de programación LISP son las expresiones simbólicas (***s-expressions***). Tanto programa como datos son representados como *s-expressions.* Una *s-expressions* puede ser ya sea un átomo o una lista.

Los átomos son las unidades sintácticas básicas, pueden ser números o símbolos:

- Los **átomos numéricos** o simplemente números, pueden ser enteros, racionales, reales y complejos, tanto positivos como negativos.
  - `12  2.5  -30  0.005  -8.0  2010  2/3 #C(0 2)  -5/8`
- Los **átomos simbólicos** o simplemente símbolos, pueden incluir palabras, números, letras y caracteres especiales.
  - `Hola  P200  Esto_es_un_atomo  +  -  /  *`

5.3.2 Expresiones Simbólicas: Listas

Una lista es una secuencia de átomos o listas separados por espacios en blanco y encerrados entre paréntesis.

```
() (9) (Nacional) (+ 12 76) (colores (rojo azul verde))
```

La lista vacía () juega un papel especial en la construcción y manipulación de las estructuras de datos en LISP y se la denomina nil. Es la única *s-expressions* que es considerada tanto como átomo y como una lista. Las listas son herramientas muy flexibles para la construcción de expresiones y estructuras.

Desde la perspectiva sintáctica, en las expresiones simbólicas, LISP no diferencia entre mayúsculas y minúsculas.

5.3.3 Control de la Evaluación en LISP

La interacción con LISP es muy similar a la que se realiza cuando se emplea una calculadora de bolsillo. En primer lugar, ante el símbolo del intérprete (>) se ingresa una expresión, luego LISP lee la expresión, la evalúa y entrega el resultado. Para prevenir la evaluación de una expresión, se debe poner antes de la expresión el apóstrofe ‘(a b c).

Si se ingresa el nombre de un símbolo sin el apóstrofe, el sistema retorna el valor asociado al símbolo si está definido, caso contrario despliega un mensaje de error.

5.3.4 Expresiones en LISP

La estructura básica, de mayor importancia en LISP, es la lista. Los siguientes son usos de listas para representar expresiones evaluables:

- `(+ 7 9)`
- `(- (+ 3 4) 6)`
- `(* (+ 2 5) (- 7 (/ 21 7)))`
- `(= (+ 2 3) 5)`
- `(> (* 5 6) (+ 4 5))`
- `(/ #C(0 2) #C(3 4))`
- `(* #C(0.5 3.7) #C(1.9 7.4))`

5.3.5 Estructuras en LISP

Una lista puede utilizarse para representar una estructura de árbol. La siguiente lista representa la descripción del árbol de la figura:

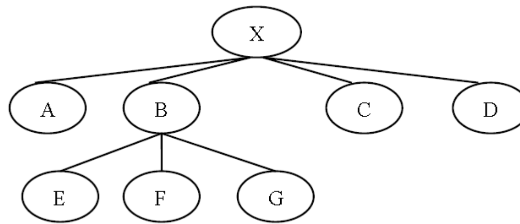


Figura 23. Representación de un Árbol

```
➢ '(X (A) (B (E) (F) (G)) (C) (D) )
```

El apóstrofe antes del paréntesis indica al intérprete LISP que la expresión debe ser considerada dato.

5.3.6 Programas en LISP

Las listas también se emplean en procedimientos y funciones. A LISP se lo puede considerar como un lenguaje de programación funcional porque utiliza funciones para manipular las estructuras de expresiones simbólicas.

```
➢ (SETF felinos '(tigre gato pantera leon))
➢ (FIRST felinos)
➢ (REST felinos)
➢ (FIRST (REST felinos))
➢ (defun cuadrado (x)
        (* x x))
➢ (defun hipotenusa (a b)
        (sqrt (+ (cuadrado a) (cuadrado b))))
➢ (defun valor-absoluto (x)
        (if (< x 0) (- x) x))
```

5.3.7 Implementaciones de LISP

Durante sus más de 50 años de vida del lenguaje LISP, ha experimentado muchas variaciones. Más aún, cada dialecto (LISP, Stanford LISP, MacLisp, InterLisp, Franz Lisp, XLISP, AutoLISP, ZetaLisp, LeLisp, Common Lisp, Dylan, Scheme, EuLisp, ISLISP, ANSI Common Lisp, ACL2) ha tenido varias implementaciones. Las diferencias entre dialectos pueden ser muy visibles, por ejemplo *Common Lisp* y *Scheme* usan diferentes palabras clave para definir funciones. Sin embargo, dentro de un dialecto estandarizado, las implementaciones que lo conforman, soportan el mismo núcleo del lenguaje, pero con diferentes extensiones y bibliotecas. Para mayor información se recomienda revisar ***The Common Lisp Directory***[108].

---

[108] http://www.cl-user.net/asp/ng9z/sdataQGAXWu7XOeEKDM==/sdataQo5Y-1Mh9urk

5.4 Representación de Problemas en el Espacio de Estados

El **espacio de estados** es un tipo de red semántica que facilita la representación de problemas. Los nodos corresponden a los estados (soluciones parciales del problema) y los enlaces son los pasos que se siguen en el proceso de solución del problema. La raíz del gráfico constituye uno o más estados iniciales dados por las condiciones del problema. También se define en el gráfico una o más metas posibles que representan las soluciones a las instancias del problema. La solución del problema está representada por el proceso de búsqueda en el espacio de estados de una ruta desde un estado inicial hasta una meta establecida.

La meta buscada puede ser un estado, una configuración final de estados o una ruta. En cualquiera de estos casos, el proceso de solución requiere explorar los diferentes enlaces partiendo de la raíz, navegando a través del gráfico, hasta que se encuentre un estado que satisfaga la meta o hasta que se hayan agotado todas las opciones. La generación de estados a ser visitados dentro de la red, se realiza mediante la aplicación de **operadores**, siguiendo alguna estrategia determinada. De esto se encarga el **algoritmo de búsqueda**.

Un algoritmo de búsqueda además de comprobar la validez de los estados que se van visitando y mantener el registro de las diferentes rutas seguidas, para escoger la mejor, debería evitar volver a explorar estados ya visitados y quedar atrapado en lazos infinitos.

Tomando el juego de **Tres en Raya**[109] como ejemplo, existe un número finito de acciones que cada jugador puede realizar. Iniciando con una matriz en blanco, el primer jugador puede poner una X en uno de los nueve espacios disponibles, el segundo jugador tiene ocho espacios para escoger y poner una O. A medida que avanza el juego cada jugador tiene un espacio menos para decidir dónde colocar su signo. Dada esta colección de posibles movimientos y respuestas, se puede representar gráficamente a cada configuración de la matriz, como un nodo o estado. Los enlaces representan los movimientos posibles que llevan a la matriz de una configuración a otra. La estructura resultante es el gráfico de un espacio de estados.

La representación de un problema como un espacio de estados, permite apreciar su complejidad. Por ejemplo, en el juego de tres en raya, a pesar de que existen 3^9 = 19.683 posibles formas de acomodar espacios, Xs y Os en la matriz de nueve celdas, muchas de ellas nunca ocurrirán en un determinado juego. Igualmente, pueden describirse 9! = 362.880 diferentes enlaces entre estados del juego. El espacio de estados resultante es un grafo antes que un árbol ya que los estados del nivel tres o más profundos, pueden ser alcanzados por varios caminos. Sin embargo no existen ciclos en este espacio de estados ya por las reglas del juego no se pueden deshacer movimientos. Un grafo con esta propiedad, se denomina **grafo direccionado acíclico**.

[109] http://www.jugargratis.org/juego/319/3-en-raya

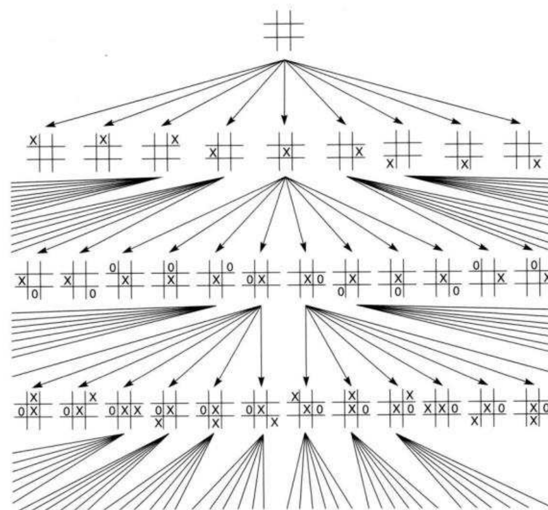


Figura 24. Espacio de Estados para el Juego Tres en Raya

Dada una representación, el siguiente componente de la solución inteligente del problema es aplicar un proceso de búsqueda apropiado a su estructura y características.

5.5 Algoritmos de Búsqueda No Informada

Este grupo de algoritmos básicamente tienen codificada cierta estrategia de navegación en el espacio de estados, pero su simplicidad hace que se caractericen porque:

- Aplican los operadores definidos en el orden en que hayan sido codificados.
- No tienen capacidad para decidir cuál es el mejor operador que puede ser aplicado en caso de haber varias opciones.
- No tiene memoria para recordar estados visitados anteriormente y pueden repetirlos en su ejecución.
- Existe una alta probabilidad de que queden atrapados en lazos infinitos.
- No son computacionalmente eficientes.

A continuación se presenta el pseudocódigo de 3 algoritmos de búsqueda no informada.

5.5.1 Algoritmo Generación y Prueba

```
Generación y Prueba
MIENTRAS meta no encontrada O no más estados EJECUTAR
     INICIO
     Generar un estado.
          SI estado igual meta ENTONCES
          RETORNAR meta encontrada
     FIN.
FIN.
```

5.5.2 Algoritmo Primero en Amplitud

```
Crear una lista Λ e inicializarla con el nodo raíz.
MIENTRAS meta no encontrada EJECUTE
INICIO /* Lazo Principal */
   SI Λ está vacía ENTONCES
      RETORNAR meta no encontrada.
```

```
Remover el primer elemento de Λ y llamarlo n.
PARA operador igual 1 HASTA NOps HACER
INICIO /* Lazo de generación de estados */
   SI operador se aplica a n ENTONCES
   INICIO
      Generar nuevo_estado.
      SI nuevo_estado igual a meta ENTONCES
         RETORNAR meta encontrada.
      Añadir nuevo_estado al final de Λ.
   FIN.
   Tomar siguiente operador
FIN. /* Lazo de generación de estados */
FIN. /* Lazo Principal */
```

5.5.3 Algoritmo Primero en Profundidad

```
Inicializar n a nodo_raíz.
ALG_Profundidad(n)
INICIO /* Algoritmo Primero en Profundidad */
   SI n igual a meta ENTONCES
      RETORNAR meta encontrada
   MIENTRAS meta  no encontrada EJECUTE
   INICIO /* Lazo Principal */
      PARA operador  igual 1 HASTA NOps HACER
      INICIO /* Lazo de generación de estados */
         SI operador  se aplica a n ENTONCES
         INICIO
            Generar estado_sucesor.
            SI no hay estado_sucesor ENTONCES
               RETORNAR meta no encontrada.
            ALG_Profundidad(estado_sucesor)
         FIN.
         Tomar siguiente operador
      FIN. /* Lazo de generación de estados */
   FIN. /* Lazo Principal */
FIN. /* Algoritmo Primero en Profundidad */
```

5.6 Procesos Heurísticos de Búsqueda

Muchos problemas de la vida real exhiben complejidad factorial o exponencial, lo cual representaría una tarea difícil o imposible para que pueda realizar una búsqueda exhaustiva de soluciones. Por ejemplo el ajedrez puede tener hasta unos 10^{120} posibles pasos de juego y damas unos 10^{40}, muchos de los cuales posiblemente nunca podrían ocurrir. Las estrategias para navegación en este tipo de espacios, requieren de técnicas heurísticas para reducir la complejidad de la búsqueda.

La técnica *heurística* mejora la eficiencia del proceso de búsqueda sacrificando exhaustividad. Las consideraciones que sirven de soporte para confiar en los procesos heurísticos, son:

- Rara vez se requieren soluciones óptimas.
- Raras veces aparecen los peores casos.
- La comprensión de un heurístico conduce a una mejor comprensión del problema.

5.6.1 Ascenso a Colina: Algoritmo de Ascenso Pronunciado

```
Inicializar estado_actual a nodo_raíz.
MIENTRAS meta no encontrada o hasta que iteración completa no
produzca cambio en estado_actual:
      Hallar el primer sucesor de estado_actual.
      Para cada operador aplicable a estado_actual
            Generar un nuevo_estado.
            SI nuevo_estado igual meta RETORNAR.
            SI nuevo_estado mejor que sucesor ENTONCES sucesor
            igual a nuevo_estado.
      SI sucesor mejor que estado_actual ENTONCES estado_actual
      igual a sucesor.
```

5.6.2 Ascenso a Colina: Algoritmo de Recocido Simulado

```
Inicializar estado_actual y mejor_estado con nodo_raíz.
Inicializar T según programa de recocido.
MIENTRAS meta no encontrada O haya nuevo operador aplicable al
estado_actual:
      Generar un nuevo_estado.
      SI nuevo_estado igual meta RETORNAR
      Calcular D=|valor estado_actual – valor nuevo_estado|.
      SI nuevo_estado mejor que estado_actual ENTONCES
      estado_actual igual nuevo_estado Y mejor_estado igual
      nuevo_estado.
      SINO
      estado_actual igual nuevo_estado,con probabilidad = e^(-D/T).
      Revisar T según programa de recocido.
Retornar mejor_estado.
```

5.6.3 Primero el Mejor: Algoritmo A*

```
Inicializar ΛA con el nodo_raíz, g = 0, f' = h'(nodo_raíz).
Inicializar ΛB como lista vacía.
MIENTRAS no se encuentre meta:
      SI ΛA está vacío, RETORNAR meta no encontrada.
      Quitar de ΛA nodo con menor valor f', llamarle mejor_nodo y
      colocarlo en ΛB.
      SI mejor_nodo igual a meta RETORNAR.
      MIENTRAS haya operadores aplicables a mejor_nodo:
            Generar un sucesor y apuntar su enlace anterior a
            mejor_nodo.
            Calcular g(sucesor) = g(mejor_nodo)+ costo(mejor_nodo a
            sucesor).
            SI sucesor igual nodo_abierto que ya está en ΛA
            ENTONCES:
                  SI g(nodo_abierto) ≤ g(sucesor) desechar sucesor
                  y apuntar un enlace posterior de mejor_nodo a
                  nodo_abierto.
                  SINO apuntar un enlace posterior de mejor_nodo a
                  nodo_abierto, el enlace anterior de nodo_abierto
                  a mejor_nodo, actualizar valores de
                  g(nodo_abierto) y f'(nodo_abierto).
            SI sucesor igual nodo_abierto que ya está en ΛB
            ENTONCES:
                  SI g(nodo_abierto) ≤ g(sucesor) desechar sucesor
                  y apuntar un enlace posterior de mejor_nodo a
                  nodo_abierto.
                  SINO apuntar un enlace posterior de mejor_nodo a
                  nodo_abierto, el enlace anterior de nodo_abierto
                  a mejor_nodo, actualizar valores de
```

```
                g(nodo_abierto), f'(nodo_abierto) y
                propagar_mejoras(nodo_abierto, g(nodo_abierto)).
            SINO colocar sucesor en ΛA, apuntar un enlace posterior
            de mejor_nodo a sucesor y calcular f'(sucesor) =
            g(sucesor) + h'(sucesor)

propagar_mejoras(n, g(n))
    MIENTRAS n tenga sucesor Y g(sucesor) < g(n):
        Calcular g(sucesor) = g(n) + costo (n a sucesor).
        Calcular f'(sucesor) = g(sucesor)+h'(sucesor).
        SI enlace anterior de sucesor A n es falso ENTONCES
        apuntar el enlace anterior de sucesor a n.
        Propagar_mejoras(sucesor, g(sucesor)).
```

Efecto de la función g:

- Si *g* = 0, el algoritmo trata de encontrar una solución, de cualquier manera.
- Si *g* = 1, el algoritmo elige el sendero con el menor número de pasos posible.
- Si *g* tiene diferentes valores para cada operador, entonces el algoritmo encuentra el sendero de menor costo posible.

Efecto de la Función *h'*

- Si *h'* = *h* (estimador perfecto) entonces A* converge inmediatamente hacia la meta (algoritmo determinístico).
- Si *h'* = 0 y *g* = 0, la estrategia de búsqueda es aleatoria.
- Si *h'* = 0 y *g* = 1, la búsqueda será primero en amplitud (algoritmo admisible pero no eficiente).

TEOREMA DE ADMISIBILIDAD:

Si *h'* nunca sobrestima a *h* entonces el algoritmo A* es admisible. La admisibilidad implica que se encontrará la solución óptima.

COROLARIO:

Si *h'* raramente sobrestima a *h* en un valor δ, entonces el algoritmo A* raramente encuentra una solución cuyo costo sea mayor en δ al costo de la solución óptima.

5.6.4 Primero el Mejor: Algoritmo Guiado por Agenda

```
MIENTRAS meta no alcanzada O agenda no agotada EJECUTAR:
    Elegir y ejecutar la tarea más prometedora de la agenda.
    Para cada uno de sus nodos sucesores:
        Calcular el valor de sucesor.tarea, combinando la
        evidencia de todas sus justificaciones.
        SI sucesor está en agenda ENTONCES
            SI sucesor.justificación no está en lista de
            justificaciones ENTONCES añadir
            sucesor.justificación a lista.
            Actualizar valor de sucesor.tarea.
        Insertar sucesor en agenda, según prioridad dada por valor
        sucesor.tarea.
```

5.7 Problema de Ejemplo en LISP

Se tiene una red de 10 nodos interconectados como se indica en la siguiente figura. El objetivo es encontrar la mejor trayectoria que comunique el nodo X0Y0, con el nodo X2Y0.

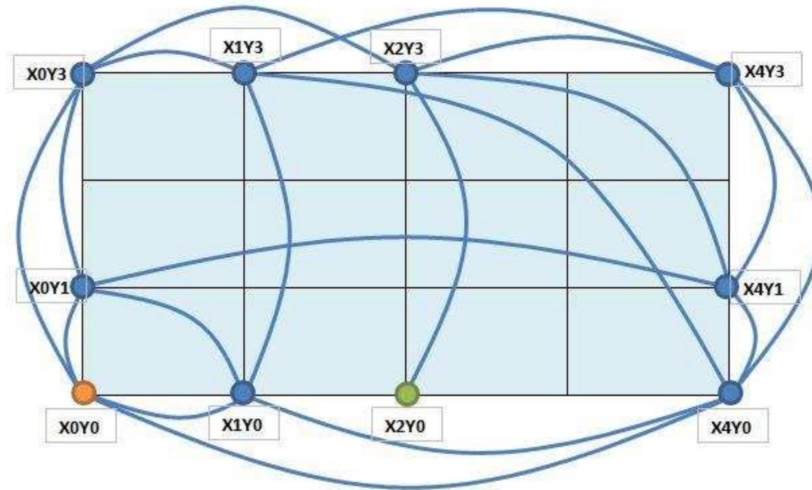


Figura 25. Grafo de 10 nodos interconectados en una red X4Y3.

Para la solución de este problema se utiliza la implementación denominada ***Allegro Common Lisp Free Express Edition***[110]. El ejemplo ilustra el poder del lenguaje LISP para la representación y el procesamiento simbólico.

En primer lugar se realizará la descripción simbólica de la topología del grafo de los 10 nodos interconectados:

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; VERSIONES ESCRITAS EN COMMON LISP (Red-X4Y3.lsp)
;; Inteligencia Artificial
;; Dr. Hugo A. Banda Gamboa
;; Dep. Informática y Ciencias de Computación
;; ESCUELA POLITÉCNICA NACIONAL
;; Quito, 29 Julio 2011
;; TOPOLOGÍA DE LA RED X4Y3
;; Para Ejecutar: load Red-X4Y3.lsp
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; La DEFINICIÓN de la TOPOLOGÍA de la RED es la siguiente:
;; a) Nodos contectados
(setf (get 'X0Y0 'conectado) '(X4Y0 X0Y3 X1Y0 X0Y1)
      (get 'X4Y0 'conectado) '(X4Y3 X0Y0 X1Y3 X4Y1 X1Y0)
      (get 'X1Y3 'conectado) '(X4Y3 X0Y3 X1Y0 X4Y0)
      (get 'X1Y0 'conectado) '(X4Y0 X1Y3 X0Y0 X0Y1)
      (get 'X0Y1 'conectado) '(X4Y1 X0Y3 X0Y0 X1Y0)
      (get 'X4Y1 'conectado) '(X4Y3 X0Y1 X4Y0 X2Y3)
      (get 'X2Y3 'conectado) '(X4Y3 X0Y3 X2Y0 X4Y1)
      (get 'X2Y0 'conectado) '(X2Y3)
      (get 'X4Y3 'conectado) '(X4Y0 X4Y1 X1Y3 X2Y3)
      (get 'X0Y3 'conectado) '(X0Y0 X0Y2 X1Y3 X2Y3))
;; b) Coordenadas de los nodos
(setf (get 'X0Y0 'coord) '(0 0)
      (get 'X0Y1 'coord) '(0 1)
      (get 'X0Y2 'coord) '(0 2)
      (get 'X0Y3 'coord) '(0 3)
      (get 'X1Y0 'coord) '(1 0)
```

110 http://www.franz.com/downloads.lhtml

```
      (get 'X1Y1 'coord) '(1 1)
      (get 'X1Y2 'coord) '(1 2)
      (get 'X1Y3 'coord) '(1 3)
      (get 'X2Y0 'coord) '(2 0)
      (get 'X2Y1 'coord) '(2 1)
      (get 'X2Y2 'coord) '(2 2)
      (get 'X2Y3 'coord) '(2 3)
      (get 'X3Y0 'coord) '(3 0)
      (get 'X3Y1 'coord) '(3 1)
      (get 'X3Y2 'coord) '(3 2)
      (get 'X3Y3 'coord) '(3 3)
      (get 'X4Y0 'coord) '(4 0)
      (get 'X4Y1 'coord) '(4 1)
      (get 'X4Y2 'coord) '(4 2)
      (get 'X4Y3 'coord) '(4 3))
```

A continuación se codifican tanto los algoritmos de búsqueda no informada (amplitud y profundidad) así como los de búsqueda informada (ascenso y primero-el-mejor):

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; VERSIONES ESCRITAS EN COMMON LISP (AlgoritmosBusq-Red.lsp)
;; Inteligencia Artificial
;; Dr. Hugo A. Banda Gamboa
;; Dep. Informática y Ciencias de Computación
;; ESCUELA POLITÉCNICA NACIONAL
;; Quito, 29 Julio 2011
;; Algoritmos de búsqueda primero en amplitud,
;; primero en profundidad, ascenso a colina y primero el mejor.
;; Para Ejecutar: load AlgoritmosBusq-Red.lsp
;; (amplitud 'X0Y0 'X2Y0) (profundidad 'X0Y0 'X2Y0)
;; (ascenso 'X0Y0 'X2Y0)  (primero-el-mejor 'X0Y0 'X2Y0)
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Definición de la función que corresponde al
;; Algoritmo de Búsqueda Primero en Amplitud.
(defun amplitud (nodo-inicial nodo-final &optional
                    (cola (list (list nodo-inicial))))
   (cond ((endp cola) nil)
         ((eq nodo-final (first (first cola)))
          (reverse (first cola)))
         (t (amplitud nodo-inicial nodo-final
              (append (rest cola) (extiende (first cola))))))
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Definición de la función que corresponde al
;; Algoritmo de Búsqueda Primero en Profundidad.
(defun profundidad (nodo-inicial nodo-final &optional
                    (cola (list (list nodo-inicial))))
   (cond ((endp cola) nil)
         ((eq nodo-final (first (first cola)))
          (reverse (first cola)))
         (t (profundidad nodo-inicial nodo-final
               (append (extiende (first cola)) (rest cola))))))
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Definición de la función que corresponde al
;; Algoritmo de Búsqueda Ascenso a Colina.
(defun ascenso (nodo-inicial nodo-final &optional
                    (cola (list (list nodo-inicial))))
   (cond ((endp cola) nil)
         ((eq nodo-final (first (first cola)))
          (reverse (first cola)))
         (t (ascenso nodo-inicial nodo-final
              (append (sort (extiende (first cola))
                 #'(lambda (p1 p2)
                      (cercanop p1 p2 nodo-final)))
                          (rest cola))))))
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Definición de la función que corresponde al
;; Algoritmo de Búsqueda Primero el Mejor.
(defun primero-el-mejor (nodo-inicial nodo-final &optional
   (cola (list (list nodo-inicial))))
   (cond ((endp cola) nil)
         ((eq nodo-final (first (first cola)))
          (reverse (first cola)))
```

```
         (t (primero-el-mejor nodo-inicial nodo-final
             (merge 'list
              (sort (extiende (first cola))
               #'(lambda (p1 p2) (cercanop p1 p2 nodo-final)))
                  (rest cola)
               #'(lambda (p1 p2) (cercanop p1 p2 nodo-final)))))))
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Definición de la función auxiliar EXTIENDE
;; cuyo objetivo es extender una trayectoria.
(defun extiende (trayectoria)
   (print (reverse trayectoria))
   (mapcar #'(lambda (nuevo-nodo) (cons nuevo-nodo trayectoria))
     (remove-if #'(lambda (conectado) (member conectado trayectoria))
       (get (first trayectoria) 'conectado))))
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Definición de la función auxiliar DISTANCIA-ENTRE-NODOS.
;; Retorna la distancia Euclidiana entre 2 nodos.
(defun distancia-entre-nodos (nodo1 nodo2)
   (let ((coord1 (get nodo1 'coord))
         (coord2 (get nodo2 'coord)))
      (sqrt (+ (expt (- (first coord1)
                        (first coord2)) 2)
               (expt (- (second coord1)
                        (second coord2)) 2)))))
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Definición de la función auxiliar CERCANOP.
;; Retorna VERDADERO si trayecto1 contiene el nodo
;; más cercano al nodo de destino.
(defun cercanop (trayecto1 trayecto2 nodo-destino)
   (< (distancia-entre-nodos (first trayecto1) nodo-destino)
      (distancia-entre-nodos (first trayecto2) nodo-destino)))
```

Las siguientes figuras muestran los resultados de la ejecución de los 4 algoritmos de búsqueda en la Red X4Y3.

```
CG-USER(1): (amplitud 'X0Y0 'X2Y0)

(X0Y0)
(X0Y0 X4Y0)
(X0Y0 X0Y3)
(X0Y0 X1Y0)
(X0Y0 X0Y1)
(X0Y0 X4Y0 X4Y3)
(X0Y0 X4Y0 X1Y3)
(X0Y0 X4Y0 X4Y1)
(X0Y0 X4Y0 X1Y0)
(X0Y0 X0Y3 X0Y2)
(X0Y0 X0Y3 X1Y3)
(X0Y0 X0Y3 X2Y3)
(X0Y0 X1Y0 X4Y0)
(X0Y0 X1Y0 X1Y3)
(X0Y0 X1Y0 X0Y1)
(X0Y0 X0Y1 X4Y1)
(X0Y0 X0Y1 X0Y3)
(X0Y0 X0Y1 X1Y0)
(X0Y0 X4Y0 X4Y3 X4Y1)
(X0Y0 X4Y0 X4Y3 X1Y3)
(X0Y0 X4Y0 X4Y3 X2Y3)
(X0Y0 X4Y0 X1Y3 X4Y3)
(X0Y0 X4Y0 X1Y3 X0Y3)
(X0Y0 X4Y0 X1Y3 X1Y0)
(X0Y0 X4Y0 X4Y1 X4Y3)
(X0Y0 X4Y0 X4Y1 X0Y1)
(X0Y0 X4Y0 X4Y1 X2Y3)
(X0Y0 X4Y0 X1Y0 X1Y3)
(X0Y0 X4Y0 X1Y0 X0Y1)
(X0Y0 X0Y3 X1Y3 X4Y3)
(X0Y0 X0Y3 X1Y3 X1Y0)
(X0Y0 X0Y3 X1Y3 X4Y0)
(X0Y0 X0Y3 X2Y3 X4Y3)
(X0Y0 X0Y3 X2Y3 X2Y0)
CG-USER(2):
```

```
CG-USER(2): (profundidad 'X0Y0 'X2Y0)

(X0Y0)
(X0Y0 X4Y0)
(X0Y0 X4Y0 X4Y3)
(X0Y0 X4Y0 X4Y3 X4Y1)
(X0Y0 X4Y0 X4Y3 X4Y1 X0Y1)
(X0Y0 X4Y0 X4Y3 X4Y1 X0Y1 X0Y3)
(X0Y0 X4Y0 X4Y3 X4Y1 X0Y1 X0Y3 X0Y2)
(X0Y0 X4Y0 X4Y3 X4Y1 X0Y1 X0Y3 X1Y3)
(X0Y0 X4Y0 X4Y3 X4Y1 X0Y1 X0Y3 X1Y3 X1Y0)
(X0Y0 X4Y0 X4Y3 X4Y1 X0Y1 X0Y3 X2Y3)
(X0Y0 X4Y0 X4Y3 X4Y1 X0Y1 X0Y3 X2Y3 X2Y0)
CG-USER(3): (ascenso 'X0Y0 'X2Y0)

(X0Y0)
(X0Y0 X1Y0)
(X0Y0 X1Y0 X4Y0)
(X0Y0 X1Y0 X4Y0 X4Y1)
(X0Y0 X1Y0 X4Y0 X4Y1 X0Y1)
(X0Y0 X1Y0 X4Y0 X4Y1 X0Y1 X0Y3)
(X0Y0 X1Y0 X4Y0 X4Y1 X0Y1 X0Y3 X0Y2)
(X0Y0 X1Y0 X4Y0 X4Y1 X0Y1 X0Y3 X2Y3)
(X0Y0 X1Y0 X4Y0 X4Y1 X0Y1 X0Y3 X2Y3 X2Y0)
```

```
CG-USER(4): (primero-el-mejor 'X0Y0 'X2Y0)

(X0Y0)
(X0Y0 X1Y0)
(X0Y0 X1Y0 X4Y0)
(X0Y0 X4Y0)
(X0Y0 X4Y0 X1Y0)
(X0Y0 X4Y0 X1Y0 X0Y1)
(X0Y0 X4Y0 X1Y0 X0Y1 X4Y1)
(X0Y0 X4Y0 X4Y1)
(X0Y0 X4Y0 X4Y1 X0Y1)
(X0Y0 X4Y0 X4Y1 X0Y1 X1Y0)
(X0Y0 X1Y0 X4Y0 X4Y1)
(X0Y0 X1Y0 X4Y0 X4Y1 X0Y1)
(X0Y0 X1Y0 X0Y1)
(X0Y0 X1Y0 X0Y1 X4Y1)
(X0Y0 X1Y0 X0Y1 X4Y1 X4Y0)
(X0Y0 X0Y1)
(X0Y0 X0Y1 X1Y0)
(X0Y0 X0Y1 X1Y0 X4Y0)
(X0Y0 X0Y1 X1Y0 X4Y0 X4Y1)
(X0Y0 X0Y1 X4Y1)
(X0Y0 X0Y1 X4Y1 X4Y0)
(X0Y0 X0Y1 X4Y1 X4Y0 X1Y0)
(X0Y0 X0Y1 X4Y1 X2Y3)
(X0Y0 X0Y1 X4Y1 X2Y3 X2Y0)
CG-USER(5):
```

Figura 26. Resultados de la ejecución de los algoritmos en la Red X4Y3

5.8 Problemas Propuestos

5.8.1 Los 2 Recipientes de Agua

Se tienen 2 recipientes, uno de 4 galones y otro de 3 galones. Ninguno tiene marcas intermedias de nivel. Existe una fuente, de la cual se puede llenar los recipientes con agua. Indicar, ¿Cómo se puede tener exactamente 2 galones de agua en el recipiente de 4 galones y 0 galones en el de 3? Las variables y sus posibles estados, son: Recipiente de 4 galones (x = 0, 1, 2, 3, 4), recipiente de 3 galones (y = 0, 1, 2, 3).

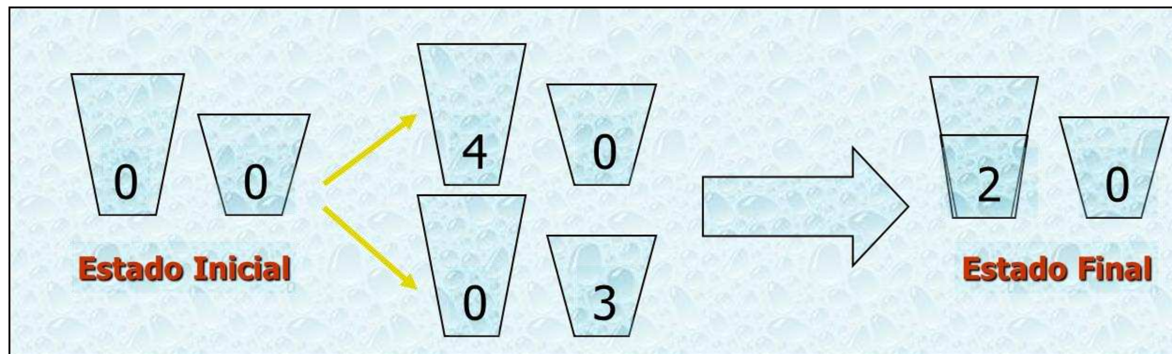


Figura 27. Problema de los 2 recipientes de agua

Los operadores definidos son los siguientes:

```
1. Llenar el recipiente de 4 galones:
        SI x < 4  →  (4, y)
2. Llenar el recipiente de 3 galones:
        SI y < 3  →  (x, 3)
3. Vaciar el recipiente de 4 galones:
        SI x > 0  →  (0, y)
4. Vaciar el recipiente de 3 galones:
        SI y > 0  →  (x, 0)
5. Pasar agua del recipiente de 3 al de 4, hasta que se llene:
        SI (x + y) ≥ 4 ^ y > 0 → (4, y - (4 - x))
6. Pasar agua del recipiente de 4 al de 3, hasta que se llene:
        SI (x + y) ≥ 3 ^ x > 0 → (x - (3 - y), 3)
7. Pasar toda el agua del recipiente de 3 galones al de 4:
        SI (x + y) ≤ 4 ^ y > 0 → (x + y, 0)
8. Pasar toda el agua del recipiente de 4 galones al de 3:
        SI (x + y) ≤ 3 ^ x > 0 → (0, x + y)
```

5.8.2 División de la Jarra de Vino en 2 Partes Iguales

Se tiene una jarra de vino de 8 litros y dos recipientes vacíos, uno de 3 y otro de 5 litros, pero sin marcas intermedias. El problema es dividir el vino de la jarra en 2 recipientes, de tal manera que cada uno tenga exactamente 4 litros. Las variables y sus posibles estados, son: Recipiente de 8 litros (x = 0, 1, 2, 3, 4, 5, 6, 7, 8); recipiente de 3 litros (y = 0, 1, 2, 3); y, recipiente de 5 litros (z= 0, 1, 2, 3, 4, 5).

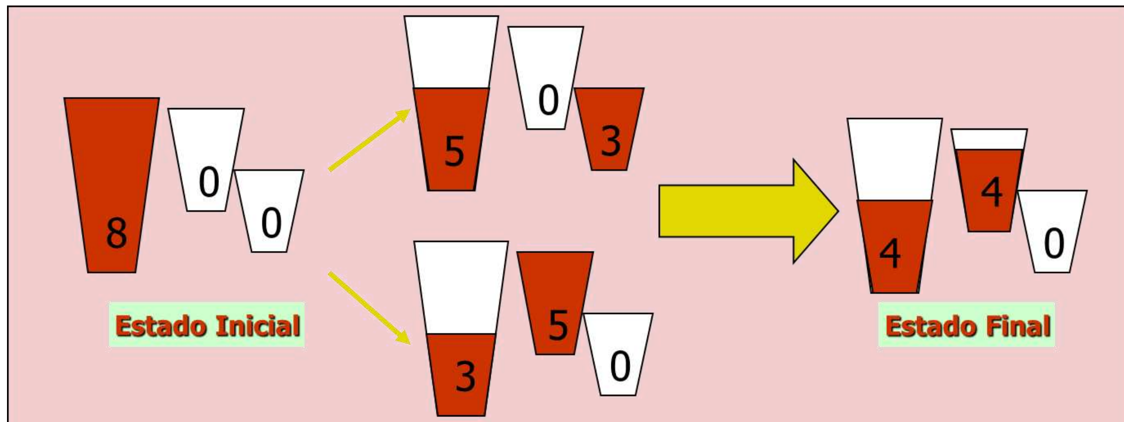


Figura 28. División de la Jarra de Vino en 2 partes iguales

Los operadores definidos son los siguientes:

- Pasar vino del recipiente de 8 al de 3, hasta que se llene:
  $SI\ (x + y) \geq 3 \wedge y < 3 \rightarrow [x - (3 - y), 3, z]$
- Pasar vino del recipiente de 8 al de 5, hasta que se llene:
  $SI\ (x + z) \geq 5 \wedge z < 5 \rightarrow [x - (5 - z), y, 5]$
- Pasar vino del recipiente de 5 al de 3, hasta que se llene:
  $SI\ (y + z) \geq 3 \wedge y < 3 \rightarrow [x, 3, z - (3 - y)]$
- Pasar vino del recipiente de 3 al de 5, hasta que se llene:
  $SI\ (y + z) \geq 5 \wedge z < 5 \rightarrow [x, y - (5 - z), 5]$
- Pasar todo el vino del recipiente de 3 al de 5:
  $SI\ (y + z) \leq 5 \wedge y > 0 \rightarrow [x, 0, z + y]$
- Pasar todo el vino del recipiente de 3 al de 8:
  $SI\ (x + y) \leq 8 \wedge y > 0 \rightarrow [x + y, 0, z]$
- Pasar todo el vino del recipiente de 5 al de 3:
  $SI\ (y + z) \leq 3 \wedge z > 0 \rightarrow [x, y + z, 0]$
- Pasar todo el vino del recipiente de 5 al de 8:
  $SI\ (x + z) \leq 8 \wedge z > 0 \rightarrow [x + z, y, 0]$

5.8.3 El Mundo de los Bloques

Ordenar en forma ascendente (o descendente), con el mínimo de movimientos, un cierto número de bloques, identificados por la letras del alfabeto, que inicialmente están apilados en forma desordenada.

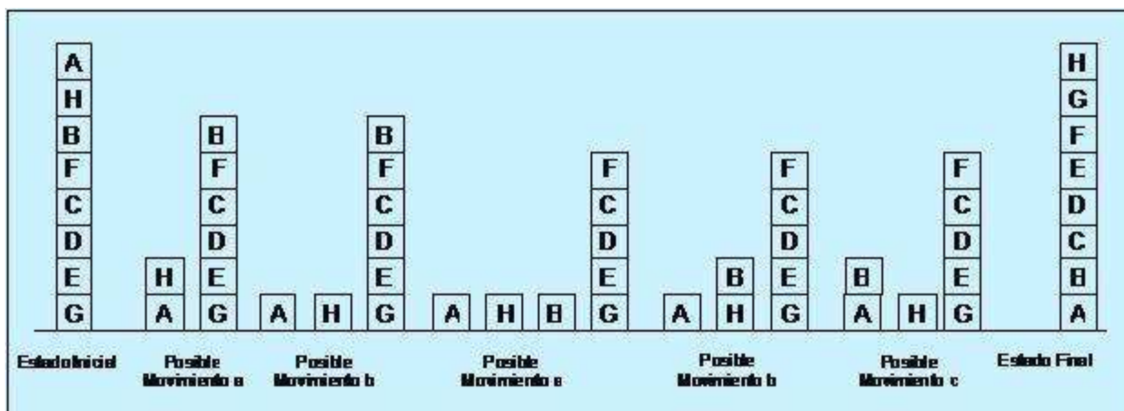


Figura 29. El Mundo de los Bloques

FUNCIÓN HEURÍSTICA:

- Local.- Añadir un punto por cada bloque sobre el lugar correcto. Restar un punto por cada bloque sobre el lugar incorrecto.
- Global.- Para cada bloque que esté sobre la estructura de apoyo correcta, añadir un punto por cada bloque en la pila. Para cada bloque que está sobre una estructura incorrecta, restar un punto por cada bloque en la pila.

OPERADORES:

```
1. Libre(x) → Sobre(x, Mesa)
2. Libre(x) Y Libre(y) → Sobre(x, y)
```

6 CAPÍTULO VI: LÓGICA FORMAL

6.1 Introducción

La lógica formal es la parte de la lógica que se dedica al estudio de la inferencia mediante la construcción de lenguajes formales, sistemas deductivos y semánticas formales. La idea es que estas construcciones capturen las características esenciales de las inferencias válidas en los lenguajes naturales, pero que al ser estructuras formales y susceptibles de análisis matemático, permiten realizar demostraciones rigurosas sobre ellas[111].

6.2 Inteligencia Basada en Lógica Matemática

John McCarthy lideró la investigación en lógica matemática para la Inteligencia Artificial. La lógica matemática estudia los sistemas formales en relación con el modo en el que codifican conceptos intuitivos de objetos matemáticos como conjuntos, números, demostraciones y computación. La lógica matemática no es la *lógica de las matemáticas* sino la *matemática de la lógica*. Incluye aquellas partes de la lógica que pueden ser modeladas y estudiadas matemáticamente.

Este tipo de lógica parte de los razonamientos correctos conocidos para desarrollar una teoría lógica y consecuentemente, inferencias más complejas que no se utilizan normalmente en la vida cotidiana. La lógica matemática es parte de la lógica formal.

6.3 Inferencia y Razonamiento

El término razonamiento tiene dos acepciones: una **procesal** (la actividad del agente que razona) y otra **funcional** (la relación entre las premisas y la conclusión).

- De los aspectos procesales de los razonamientos se ocupa la psicología, en el caso de que el agente sea humano. Pero si el agente es un computador entonces es un asunto propio de la inteligencia artificial.
- La lógica se ocupa de los razonamientos en el sentido funcional. En el proceso que lleva de las premisas a la conclusión pueden encadenarse múltiples pasos elementales. En la lógica se estudian las condiciones bajo las cuales estos pasos son correctos, pero no cómo y en qué orden deben realizarse: se supone que la mente dispone de los mecanismos adecuados para hacerlo.

El proceso de razonamiento permite llegar a conclusiones que *pueden o no ser correctas*, a partir de un conjunto de hechos conocidos como verdaderos.

---

111 http://es.wikipedia.org/wiki/Lógica_formal

Una inferencia es simplemente un razonamiento formal, en el sentido de que lo importante es la forma de las premisas y la conclusión y la relación entre ellas, no su contenido. Inferir es sacar una consecuencia categórica, indiscutible, a partir de hechos conocidos o asumidos como verdaderos. Realizar inferencias significa derivar nuevos hechos *inobjetables*, a partir de un conjunto de hechos dados como verdaderos.

6.3.1 Reglas de Inferencia Lógica

- Modus Ponens

  Si p y la implicación ($p \rightarrow q$) son verdaderas, entonces q tiene que ser verdadera.

- Modus Tolens

  Si la implicación ($p \rightarrow q$) es verdadera y q es falsa, entonces p tiene que ser falsa.

- Resolución

  Si la unión ($p \vee q$) es verdadera y la unión ($\sim q \vee r$) es verdadera, entonces la unión ($p \vee r$) tiene que ser verdadera.

p	q	r	$p \vee q$	$\sim q \vee r$	$p \vee r$	$\sim p \vee q$	$p \rightarrow q$
F	F	F	F	V	F	V	V
F	F	V	F	V	V	V	V
F	V	F	V	F	F	V	V
F	V	V	V	V	V	V	V
V	F	F	V	V	V	F	F
V	F	V	V	V	V	F	F
V	V	F	V	F	V	V	V
V	V	V	V	V	V	V	V

Figura 30. Tabla de Verdad de Operadores de Inferencia Lógica

6.4 Métodos de Razonamiento Lógico

Los métodos de razonamiento lógico, pueden ser de tipo deductivo, abductivo o inductivo:

- **Deductivo**: Sus conclusiones son siempre correctas. Forma monotónica de realizar inferencias. Un razonamiento se llama monotónico cuando a lo largo del proceso el conjunto de hechos conocidos es siempre creciente. Pero en la realidad suele ocurrir que, a medida que avanza el proceso de inferencias, nuevas evidencias o acciones del mismo sistema anulan premisas o conclusiones anteriores, y para formalizar esto se necesita una lógica no monótona.

  Para todo p, q, r; si p mayor que q y q mayor que r, entonces p mayor que r.

- **Abductivo**: Sus conclusiones pueden o no ser correctas. Es un método para generar explicaciones. La abducción está en la base de los sistemas basados en conocimiento que razonan con una lógica bayesiana

  Si ($p \rightarrow q$) es verdadera y q es verdadera, entonces p podría ser verdadera.

- **Inductivo**: Sus conclusiones pueden o no ser correctas. Es la base de la investigación científica.

```
Si P(a), P(b), ... P(n) son verdaderos, entonces se puede concluir
que para todo x, P(x) puede ser verdadero.
```

El ejemplo de razonamiento inductivo dado es un **razonamiento por generalización**, que va de lo particular a lo general. Pero hay otros razonamientos inductivos que proceden por **analogía**. Ejemplos:

- **De lo general a lo general:**
  - Todos los gorriones son pájaros y hacen nidos
  - Todas las gaviotas son pájaros y hacen nidos
  - Todos los cuervos son pájaros
  - **Todos los cuervos hacen nidos**
- **De lo particular a lo particular:**
  - A es político y es mentiroso
  - B es político y es mentiroso
  - C es político
  - **C es mentiroso**

La **generalización inductiva** es importante en el campo de la adquisición de conocimiento mediante aprendizaje y en la minería de datos. El **razonamiento por analogía** en cambio lo es en los sistemas de conocimiento basados en casos.

6.5 Lógica de Predicados

Predicado es una sentencia que expresa relaciones entre objetos, así como también cualidades y atributos de tales objetos.

- Los objetos pueden ser personas, objetos físicos o conceptos.
- Las cualidades, relaciones o atributos, se denominan predicados.
- Los *objetos* se conocen como argumentos o términos del *predicado*.

6.5.1 Características de los Predicados

Los predicados tienen asociado un valor de veracidad que resuelve a verdadero o falso, dependiendo de sus términos o argumentos.

Los predicados pueden ser usados para asignar una cualidad abstracta a sus términos o para representar acciones o relaciones de acción entre dos objetos.

Al construir un predicado se asume que su veracidad está basada en su relación con el mundo real.

6.5.2 Predicados, Axiomas y Patrones

A los predicados cuyos argumentos representen a un objeto específico (términos constantes), que sean establecidos y asumidos como lógicamente verdaderos, se los denomina **axiomas**. A los predicados que tienen variables como argumentos, se los denomina **patrones**.

Ventajas de la Lógica de Predicados: Permite fácil representación de conocimiento declarativo.

Desventajas de la Lógica de Predicados: Dispone de sólo dos niveles de veracidad (verdadero, falso). Tiene formalismo de razonamiento monotónico (el conjunto de axiomas continuamente crece de tamaño).

6.6 PROLOG

Proviene del francés ***Pro**gramation et **Log**ique*. Es un lenguaje de programación para inteligencia artificial que hace uso de la lógica de predicados. Fue desarrollado por **Philippe Roussel** y **Alain Colmerauer** en la Universidad de Marsella en 1972. No obstante, el primer compilador lo construyó el matemático **Robert Kowalski**, del Imperial College de Londres[112]. Al principio era sólo un lenguaje interpretado; luego, a mediados de los 70, **David Warren desarrolló** un compilador que traducía Prolog a un conjunto de instrucciones de una máquina abstracta denominada ***Warren Abstract Machine*** (WAM).

Las primeras versiones del lenguaje diferían en muchos aspectos de sus sintaxis, empleándose mayormente como forma normalizada el dialecto propuesto por la **Universidad de Edimburgo**[113] hasta que en 1995 se estableció el estándar llamado ISO-Prolog (ISO/IEC 13211-1). Prolog se enmarca en el paradigma de los lenguajes lógicos y declarativos, lo que le diferencia enormemente de otros lenguajes más populares tales como Fortran, Pascal, C o Java. Prolog, Es un lenguaje de programación centrado alrededor de un pequeño conjunto de mecanismos básicos, incluyendo la unificación de patrones, estructuración de datos basado en árboles y retroceso automático. Es apropiado para problemas que involucran objetos estructurados y relaciones entre ellos.

6.6.1 Programación en Prolog

La programación está basada en la **lógica de primer orden** LPO (o **lógica de predicados**). Consiste en definir relaciones y preguntar acerca de dichas relaciones. El programa está conformado por cláusulas. Las cláusulas pueden ser de 3 tipos: hechos, reglas y preguntas.

- Una relación puede ser especificada por medio de hechos o por reglas que definen la relación. Un procedimiento es un conjunto de cláusulas acerca de una misma relación.
- Preguntar acerca de relaciones en Prolog es similar a interrogar a una base de datos (*query*). Prolog responde con un conjunto de objetos que satisfacen la pregunta.

Un programa PROLOG consiste en un conjunto de proposiciones (o sentencias), de la forma:

---

[112] Kowalski, R A. *THE EARLY YEARS OF LOGIC PROGRAMMING*. Communications of the ACM, Jan 1988, Vol. 31, N° 1, pp. 38 – 43.

[113] http://www.inf.ed.ac.uk/teaching/courses/aipp/

```
A:- B1, ... , Bn.  Donde n ≥ 0.
```

- Cuando `n > 0`, la sentencia se escribe `A :- B1, ... , Bn.` Se denomina regla.
- Si `n = 0`, la sentencia se escribe `A`. Se denomina hecho.

Un programa PROLOG consiste en un conjunto de hechos (afirmaciones simples) y de reglas que afirman:

```
El hecho A es cierto si son ciertos los hechos B1, y B2, y … y Bn.
```

Las reglas sirven para deducir nuevos hechos a partir de otros. A un programa PROLOG se le hacen preguntas. Una pregunta se escribe como:

```
?A1, A2, …, Am. Siendo m > 0.
```

Informalmente, dicha pregunta se leerá:

```
¿Son ciertos los hechos A1 y A2 y … y Am?
```

Prolog, para establecer si un objeto definido satisface una pregunta, sigue un proceso complejo que involucra inferencia lógica y un mecanismo de retroceso (*backtracking*). Esto es hecho automáticamente y está oculto al usuario. Un programa en Prolog tiene dos tipos de significados:

- Declarativo.- Asociado con las relaciones definidas en el programa.
- Procedimental.- Asociado con la forma en que Prolog evalúa las relaciones definidas, para determinar las respuestas.

6.6.1.1 Objetos de Datos

Prolog tiene como datos: objetos simples y estructuras. Esto se ilustra en la siguiente Figura.

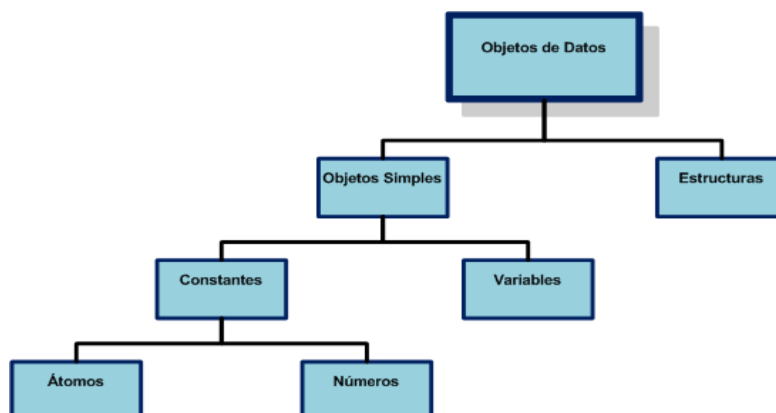


Figura 31. Objetos de Datos en Prolog

Los objetos simples pueden ser constantes o variables. Los objetos constantes, a su vez, pueden ser:

- Átomos.- Sus nombres inician con letra minúscula, o son cadenas de caracteres encerrados entre comillas:
    - `juan  ana  x_25  'Juan'  'Ana'  'Facultad de Sistemas'`
- Números.- Ya sean estos Enteros o Reales:
    - `4  1356  0  -98`
    - `3.14  -0.0035  100.2`

Los nombres de los objetos Variables, siempre inician con una letra mayúscula o con una subraya:

- `X  Juan  Ana  Resultado  _x_25  _123`

Cuando el nombre de una Variable es sólo una subraya, se denomina Variable Anónima y posee características especiales en Prolog: _

6.6.1.2 Estructuras

Las estructuras son objetos que tienen varios componentes. Sintácticamente, en Prolog, todos los objetos de datos se denominan términos. Los componentes pueden ser objetos simples o estructuras. A pesar de estar conformados por varios componentes, las estructuras en Prolog son tratadas como objetos simples.

Para combinar los componentes (argumentos) en un objeto simple, se usa un ***functor***:

- `fecha(1, marzo, 2010).`
- `fecha(Dia, Mes, 2010).`
- `punto(6, 4).`
- `punto(X, Y).`

Un *functor* está definido por:

- El nombre, cuya sintaxis es igual a la de los átomos.
- El *arity*, esto es el número de argumentos.

Todas las estructuras pueden ser visualizadas como árboles, en el que la raíz en el *functor* y sus ramificaciones son sus componentes.

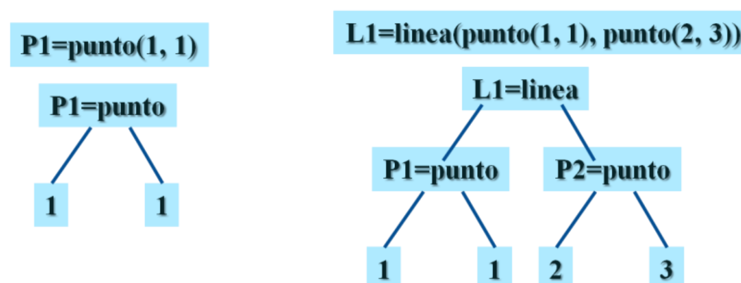


Figura 32. Predicados como estructuras de árbol en Prolog

6.6.1.3 Listas en Prolog

La lista es una estructura de datos que contiene una secuencia de cero o más objetos encerrados entre corchetes y separados por comas. Las listas pueden ser vacías o no.

- `[jugar, estudiar, viajar, coleccionar]    []`

Una lista no vacía tiene 2 partes el encabezado y la cola. El encabezado puede ser un árbol o una variable y la cola tiene que ser una lista.

- `.(Encabezado, Cola)`

La lista puede ser vista como representación de un árbol.

- `.(jugar, .(estudiar, .(viajar, .(coleccionar, [])))`

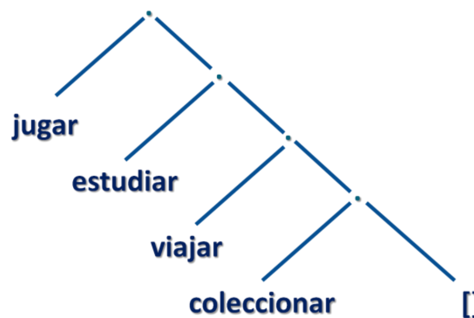


Figura 33. Lista en Prolog representada como árbol

6.6.1.4 Operaciones con Listas

Prolog proporciona la notación de la barra vertical, para facilitar la operación con listas.

- `[a, b, c]=[a|[b, c]]=[a, b|[c]]`

Un objeto X es miembro de una lista, si se cumple que:

- X encabeza la lista; o,
- X se encuentra en la Cola de la lista.

```
miembro(X,[X|Cola].

miembro(X,[Cabeza|Cola]):-

      miembro(X,Cola).
```

6.6.1.5 La Unificación

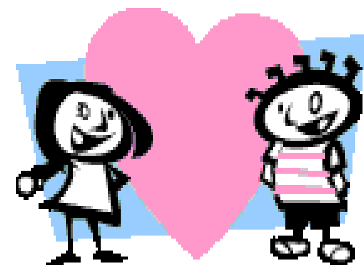
Para determinar la veracidad de sentencias compuestas por predicados y conectivos lógicos, es necesario evaluar la veracidad de cada uno de sus términos y luego aplicar los operadores lógicos a los resultados.

Un predicado componente resuelve a verdadero si se identifica con un axioma de la base de conocimiento. La unificación es un proceso que computa las sustituciones apropiadas para determinar si dos objetos coinciden o se identifican.

El Proceso de Unificación se realiza en base a las siguientes condiciones:

- Todo predicado que no contenga variables en sus argumentos, debe tener un axioma que se identifique totalmente, para considerarlo como verdadero.
- Si un predicado tiene una variable, ésta debe ser asociada a un valor determinado. Se seleccionan todos los axiomas de la base que se identifiquen con el patrón en todo excepto por la variable. La variable es asociada con el valor del argumento de la posición correspondiente del axioma.
- Si más de un axioma se identifica con el predicado dado, todos los valores asociados son considerados y tratados separadamente.
- El proceso de identificación continúa asumiendo que el valor de la variable es el valor asociado, en cualquier lugar que ésta aparezca.
- Los conectivos lógicos son aplicados a todos los predicados, para determinar la veracidad de la sentencia dada.

6.7 Problema de Ejemplo en PROLOG



Dado un conjunto de personas, hombres y mujeres, que tienen ciertos *hobbies*, se trata de establecer cuáles pueden conformar una pareja ideal, suponiendo que para ello deben compartir al menos uno de los *hobbies* declarados.

Para resolver este problema se utilizará **SWI-Prolog**[114], que es una versión de software libre (*Lesser GNU Public License*). Además, para usar **SWI-Prolog**[115] bajo **MS-Windows**, se recomienda también instalar el entorno de desarrollo **SWI-Prolog-Editor**[116].

En primer lugar se definen como axiomas al grupo de personas, donde cada una está representada por un nombre, su género y una lista de hobbies.

A continuación se codifica la regla de inferencia: Se puede decir que es una pareja ideal si existe un hombre y una mujer que comparten al menos uno de sus hobbies (interés común). Para determinar si el hombre y la mujer tienen algún interés común, se analiza el contenido de las dos listas de hobbies, buscando alguno que sea miembro tanto de la lista del hombre, como de la lista de la mujer. Finalmente, se imprime en la salida los resultados de las parejas que se consideran ideales.

---

[114] http://www.swi-prolog.org/
[115] http://www.swi-prolog.org/versions.html
[116] http://lakk.bildung.hessen.de/netzwerk/faecher/informatik/swiprolog/indexe.html

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% VERSION ESCRITA EN SWI PROLOG (ParejaIdeal.pl)
%% Inteligencia Artificial
%% Dr. Hugo A. Banda Gamboa
%% Dep. Informática y Ciencias de Computación
%% ESCUELA POLITÉCNICA NACIONAL
%% Quito, 29 Julio 2011
%% Para Ejecutar:
%%    Archivo -> Abrir ParejaIdeal.pl
%%    Iniciar -> Consultar
%%    ?- pareja_ideal.

% Axiomas
persona('Manuel',hombre,[viajar,libros,computadores,futbol]).
persona('Roberto',hombre,[estampillas,futbol,nadar,viajar]).
persona('Miriam',mujer,[vinos,libros,computadores,modas]).
persona('Jenny',mujer,[estampillas,tejer,nadar,viajar]).

% Reglas
pareja_ideal:-
        persona(H,hombre,Lh),
        persona(M,mujer,Lm),
        interes_comun(Lh,Lm),
        write(H),write(' podría hacer pareja con '), write(M), nl,fail.
miembro(X,[X|_]).
miembro(X,[_|Cola]):-
        miembro(X,Cola).
interes_comun(L1,L2):-
        miembro(I, L1),
        miembro(I, L2), !.
```

Si al consultar no da mensajes de error, se pueden plantear preguntas al intérprete y el sistema contestará, según corresponda:

```
% Pregunta: ¿Cuáles personas pueden ser una pareja ideal?
?- pareja_ideal.
Manuel podría hacer pareja con Miriam
Manuel podría hacer pareja con Jenny
Roberto podría hacer pareja con Jenny
false.

% Pregunta: ¿Cuáles son las personas y sus datos registrados?
?- persona(N,G,H).
N = 'Manuel',
G = hombre,
H = [viajar, libros, computadores, futbol] ;
N = 'Roberto',
G = hombre,
H = [estampillas, futbol, nadar, viajar] ;
N = 'Miriam',
G = mujer,
H = [vinos, libros, computadores, modas] ;
N = 'Jenny',
G = mujer,
H = [estampillas, tejer, nadar, viajar].

% Pregunta: ¿Hay mujeres registradas en el sistema?
?- persona(_,'mujer',_).
true.

% Pregunta: ¿Nombres de los hombres registrados en el sistema?
persona(N,'hombre',_).
N = 'Manuel' ;
N = 'Roberto' ;
false.
```

6.8 Problemas Propuestos

6.8.1 Dilema del Granjero

Un granjero desea cruzar un río llevando consigo un lobo, una oveja y una carga de col. El bote disponible solo permite el transporte de hasta dos objetos a la vez. Si no puede dejar solos a la oveja y al lobo o a la cabra y la carga de col, ¿cómo puede hacer para cruzar el río sin contratiempos? [117]

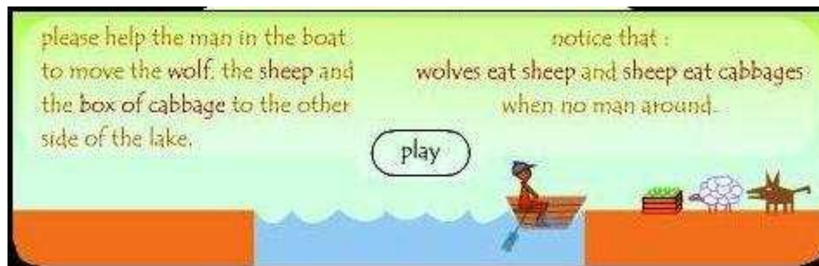


Figura 34. Dilema del Granjero

El siguiente código tomado de los ejemplos dados en la implementación de SWI Prolog, resuelve el dilema del granjero:

```
%%% Disclaimer: These programs are provided with no warranty whatsoever as to
%%% their correctness, reliability, or any other property.  We have written
%%% them for specific educational purposes, and have made no effort
%%% to produce commercial quality computer programs.  Please do not expect
%%% more of them then we have intended.
%%%
%%% This code has been tested with SWI-Prolog (Multi-threaded, Version 5.2.13)
%%% and appears to function as intended.
/*
 * This is the code for the Farmer, Wolf, Goat and Cabbage Problem
 * using the ADT Stack. Make sure that archive adts.pl is in the
 * same folder as this one: FWGC.pl
 *
 * Use the instruction: Consult All and run this code by giving PROLOG the query:
 *
 *   test.
 */
:- [adts]. /* consults (reconsults) file containing the
              various ADTs (Stack, Queue, etc.) */
go(Start,Goal) :-
        empty_stack(Empty_been_stack),
        stack(Start,Empty_been_stack,Been_stack),
        path(Start,Goal,Been_stack).
/*
 * Path predicates
 */
path(Goal,Goal,Been_stack) :-
        write('Solution Path Is:' ), nl,
        reverse_print_stack(Been_stack).
path(State,Goal,Been_stack) :-
        move(State,Next_state),
        not(member_stack(Next_state,Been_stack)),
        stack(Next_state,Been_stack,New_been_stack),
        path(Next_state,Goal,New_been_stack),!.
/*
 * Move predicates
 */
move(state(X,X,G,C), state(Y,Y,G,C))
              :- opp(X,Y), not(unsafe(state(Y,Y,G,C))),
```

117 http://www.jugargratis.org/juego/555/el-zorro-la-oveja-y-el-heno

```
                 writelist(['try farmer takes wolf',Y,Y,G,C]).
move(state(X,W,X,C), state(Y,W,Y,C))
              :- opp(X,Y), not(unsafe(state(Y,W,Y,C))),
                 writelist(['try farmer takes goat',Y,W,Y,C]).
move(state(X,W,G,X), state(Y,W,G,Y))
              :- opp(X,Y), not(unsafe(state(Y,W,G,Y))),
                 writelist(['try farmer takes cabbage',Y,W,G,Y]).
move(state(X,W,G,C), state(Y,W,G,C))
              :- opp(X,Y), not(unsafe(state(Y,W,G,C))),
                 writelist(['try farmer takes self',Y,W,G,C]).
move(state(F,W,G,C), state(F,W,G,C))
              :- writelist(['      BACKTRACK from:',F,W,G,C]), fail.
/*
 * Unsafe predicates
 */
unsafe(state(X,Y,Y,C)) :- opp(X,Y).
unsafe(state(X,W,Y,Y)) :- opp(X,Y).
/*
 * Definitions of writelist, and opp.
 */
writelist([]) :- nl.
writelist([H|T]):- print(H), tab(1),  /* "tab(n)" skips n spaces. */
                   writelist(T).
opp(e,w).
opp(w,e).
reverse_print_stack(S) :-
        empty_stack(S).
reverse_print_stack(S) :-
        stack(E, Rest, S),
        reverse_print_stack(Rest),
        write(E), nl.
test:- go(state(w,w,w,w),state(e,e,e,e)).
```

La ejecución de este programa, da como resultado:

```
test.
try farmer takes goat e w e w
try farmer takes goat w w w w
try farmer takes self w w e w
try farmer takes wolf e e e w
try farmer takes wolf w w e w
try farmer takes goat w e w w
try farmer takes goat e e e w
try farmer takes cabbage e e w e
try farmer takes wolf w w w e
try farmer takes wolf e e w e
try farmer takes goat e w e e
try farmer takes goat w w w e
try farmer takes cabbage w w e w
      BACKTRACK from: e w e e
      BACKTRACK from: w w w e
try farmer takes cabbage w e w w
try farmer takes self w e w e
try farmer takes goat e e e e
Solution Path Is:
state(w,w,w,w)
state(e,w,e,w)
state(w,w,e,w)
state(e,e,e,w)
state(w,e,w,w)
state(e,e,w,e)
state(w,e,w,e)
state(e,e,e,e)
true.
```

6.8.2 Misioneros y Caníbales

Tres misioneros y tres caníbales deben cruzar un río. El bote disponible sólo permite el transporte de hasta dos personas. Si en ningún caso deben estar más caníbales que misioneros, ¿Cuál sería el procedimiento para que todos crucen sin que haya peligro para los misioneros? [118]



Figura 35. Problema del cruce de misioneros y caníbales

[118] http://www.jugargratis.org/juego/556/misioneros-y-canibales

7 CAPÍTULO VII: SISTEMAS BASADOS EN CONOCIMIENTO

7.1 Introducción

Un Sistema Basado en Conocimiento, es un sistema computarizado que utiliza métodos y técnicas de inteligencia artificial para resolver problemas en el dominio en el cual posee conocimiento específicamente codificado. La solución es esencialmente la misma que hubiera dado un ser humano confrontado con idéntico problema, aunque no necesariamente el proceso seguido por ambos puede ser igual.

La diferencia entre los sistemas basados en conocimiento y software de aplicaciones convencionales es que estas últimas utilizan algoritmos para resolver problemas y los primeros usan conocimiento heurístico.

En general, un Sistema Basado en Conocimiento, se caracteriza por:

- Separación modular entre el conocimiento y el programa que lo procesa.
- Uso del conocimiento en un dominio específico.
- Naturaleza heurística del conocimiento utilizado.

7.2 Tipos de Conocimiento

A continuación se presenta algunas clasificaciones que se han propuesto para identificar el tipo de conocimiento:

7.2.1 Primera Clasificación

- **Conocimiento Común:** Es el que se adquiere de manera cotidiana, sin una planeación y sin la utilización de instrumentos especialmente diseñados.
- **Conocimiento Heurístico:** Se define como el conocimiento basado en la experiencia y en la percepción, que todo hombre adquiere debido a las diversas necesidades que se le presentan en la vida, adquirido muchas veces por instinto y no pensamiento fundamentado donde todo conocimiento que se genera no implica a la ciencia o leyes. Es así que existen personas con gran dominio de un determinado aspecto sin haber recibido educación alguna.
- **Conocimiento Científico:** Es un conocimiento que exige mayor rigor, que trata de encontrar las regularidades en los conocimientos para explicarlos, conocerlos y predecirlos.

7.2.2 Segunda Clasificación

- **Conocimiento Tácito:** Este es el tipo de conocimiento que permanece en un nivel "inconsciente", se encuentra desarticulado y lo implementamos y ejecutamos de una manera mecánica sin darnos cuenta de su contenido.
- **Conocimiento Intuitivo:** El conocimiento intuitivo está definido como la aprehensión inmediata de las experiencias internas o externas en su experimentación o percepción. Esto quiere decir que nos una especie de sensación vaga sobre algo que se establece como la visión clara y directa de experimentar las cosas en su forma original.
- **Conocimiento Implícito:** A diferencia del conocimiento tácito, el conocimiento implícito es el que sabemos que lo tenemos, pero no nos damos cuenta que lo estamos utilizando, simplemente lo ejecutamos y ponemos en práctica de una manera habitual.
- **Conocimiento Explícito:** El conocimiento explícito es el que sabemos que tenemos y somos plenamente conscientes cuando lo ejecutamos, es el más fácil de compartir con los demás ya que se encuentra estructurado y muchas veces esquematizado para facilitar su difusión.

7.2.3 Tercera Clasificación

- **Estratégico:** Conocimiento sobre los pasos a seguir para resolver un problema.
- **Estructural:** Sobre la organización del conocimiento del dominio, según varios niveles de abstracción. Permite organizar el conocimiento que se tiene del dominio (ejemplo las clasificaciones taxonómicas).

7.2.4 Cuarta Clasificación

- **Declarativo:** También conocido como conocimiento descriptivo, conocimiento proposicional o conocimiento de hechos, es la primera de las dos maneras en que se almacena la información en la memoria a largo plazo. El conocimiento declarativo es información consistente en hechos, conceptos o ideas conocidas conscientemente y que se pueden almacenar como proposiciones.
- **Procedimental:** Es la segunda de las dos maneras en que se almacena la información en la memoria a largo plazo. El conocimiento procedimental es el conocimiento relacionado con cosas que sabemos hacer, como por ejemplo montar en bicicleta, preparar una receta de cocina o hablar el lenguaje.
- **Semántico:** Hace referencia a los significados, entendimientos y otros conocimientos conceptuales que no están relacionados con experiencias concretas. Se considera que la recolección consciente de información sobre hechos y el conocimiento general sobre el mundo es independiente del contexto y la relevancia personal.
- **Episódico:** Se representa con pautas espacio-temporales. Sus contenidos no tienen la capacidad inferencial y se refieren a hecho únicos (incluidos los hechos autobiográficos).

7.3 Arquitectura de un Sistema Basado en Conocimiento

Un Sistema Basado en Conocimiento consiste de un conjunto de programas que configuran un motor de inferencia, la base de conocimiento, la base de hechos y las unidades de interfaz (entrada/salida).

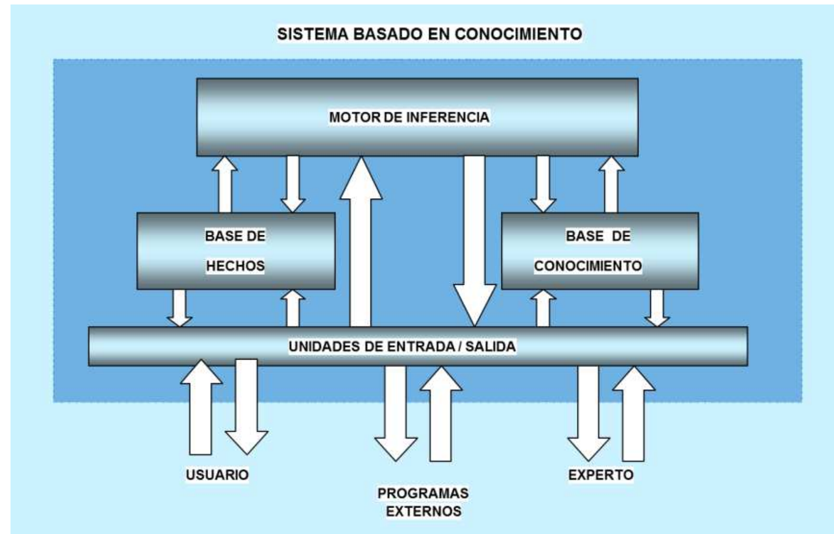


Figura 36. Arquitectura de un Sistema Basado en Conocimiento

El motor de inferencia manipula el conocimiento representado en la base de conocimiento para desarrollar una solución al problema cuyos datos están consignados en la base de hechos.

El motor de inferencia contiene el conocimiento general acerca de cómo aplicar una estrategia para resolver el problema; la base de conocimiento contiene el conocimiento codificado acerca de las particularidades del problema; y, la base de hechos contiene la información inicial del problema, así como la información que progresivamente se va derivando, a medida que se va resolviendo el problema.

Un Sistema Basado en Conocimiento puede ser visto desde tres diferentes perspectivas: la del usuario final, la del ingeniero de conocimiento y la del desarrollador de la herramienta.

- Desde la perspectiva del **usuario**, el sistema es como una caja negra con la que se comunica a través de su propia interfaz. Dependiendo de la naturaleza del sistema, el usuario tendría que ingresar en la base de hechos, ciertos datos requeridos durante la ejecución del sistema.
- Desde la perspectiva del **ingeniero de conocimiento**, él es quien desarrolla la base de conocimiento del sistema, por lo que ve mayores detalles del mismo a través de un entorno de desarrollo, especialmente diseñado para facilitar su trabajo, el mismo que incluye: un ambiente para adquisición del conocimiento, una base con datos de prueba de casos y una interfaz de programación. Para el ingeniero de conocimiento, el motor de inferencia es como si fuera, en general, una caja negra

- La perspectiva del **desarrollador de la herramienta**, es la más compleja. Se encarga de programar el motor de inferencia e incorpora las herramientas para el desarrollo del sistema. Adicionalmente, es quien decide acerca de cuan general o especializada va a ser la herramienta. Una herramienta general sirve para un amplio rango de aplicaciones, sacrificando eficiencia. En cambio, una herramienta especializada limita las aplicaciones pero puede incluir características muy atractivas para el segmento en el cual presta su contingente.

Los modelos de inteligencia artificial están muy enraizados en la lógica, porque esta proporciona formalismo para dos importantes conceptos en los sistemas basados en conocimiento: representación del conocimiento y automatización del proceso de inferencia.

7.4 Representación del Conocimiento

Existen varias formas para representar el conocimiento en forma simbólica:

- Reglas de producción
- Redes semánticas
- Plantillas o marcos (*Frames*)
- Objetos

7.4.1 Representación Mediante Reglas de Producción

Un sistema de producción está formado por un conjunto de reglas de producción almacenadas en una memoria de trabajo (correspondiente a la memoria a corto plazo) y un procesador. Cada regla de producción tiene dos partes: un patrón y una acción. El patrón es un racimo que puede emparejarse con algún elemento en la memoria de trabajo; cuando se produce el emparejamiento se dice que la regla se dispara, y entonces el procesador ejecuta la acción especificada en la regla.

Es una formulación muy general que se puede concretar de distintas maneras. Una acción puede ser una salida hacia el subsistema motor, una petición de recuperación de la memoria a largo plazo, o de almacenamiento en ella, o la generación de un nuevo patrón. Un patrón puede ser una simple declaración o proposición (como en las reglas de la Figura 1.4), o una estructura más compleja. Por otra parte, el procesador tiene que ocuparse, entre otras cosas de la resolución de conflictos que surge cuando son varias las reglas aplicables.

Las reglas de producción son importantes formas de representación del conocimiento. Las reglas de producción representan el conocimiento utilizando un formato SI-ENTONCES (IF-THEN), es decir tienen dos partes:

- SI antecedente, premisa, condición o situación es verdadero
- ENTONCES consecuente, conclusión, acción o respuesta que se ejecuta.

Los sistemas basados en reglas de producción son los más comúnmente utilizados.

7.4.1.1 Razonamiento en Sistemas Basados en Reglas de Producción

Un sistema basado en reglas de producción utiliza el *modus ponens* para manipular las *afirmaciones* y las *reglas* durante el proceso de inferencia.

Una declaración de que algo es verdadero o es un hecho conocido, es una *afirmación*. Al conjunto de *afirmaciones* se lo denomina memoria de trabajo o *Base de Hechos*; y, al conjunto de *reglas* se lo denomina *Base de Reglas*.

Mediante técnicas de búsqueda y procesos de emparejamiento (*matching*) los sistemas basados en reglas automatizan sus métodos de razonamiento.

Cuando el proceso parte considerando todos los hechos conocidos y luego avanza hacia la solución, se lo denomina guiado por los datos o de **Encadenamiento Progresivo**. Sus principales características son:

- Rápida respuesta a los cambios en los datos de entrada.
- Pocas relaciones predeterminadas entre los datos de entrada y las conclusiones derivadas.
- Las reglas expresan conocimiento como patrones generales y las conexiones precisas entre estas reglas no pueden ser predeterminadas.
- Los sistemas son adecuados para trabajar en sistemas de control de procesos en tiempo real, diseño, planeamiento y calendarización, donde los datos están continuamente siendo adquiridos, modificados, actualizados o se sintetizan nuevos hechos basados en las conclusiones de las reglas.

Cuando el proceso selecciona una posible solución y trata de probar su validez buscando evidencia que la apoye, se lo denomina guiado por el objetivo o de **Encadenamiento Regresivo**. Las principales características, son:

- Apropiados para trabajar con aplicaciones que tengan mucho mayor número de entradas, que de soluciones posibles.
- Excelentes para aplicaciones en las que el usuario dialoga directamente con el sistema y proporciona datos a través de la interfaz de entrada: Sistemas de diagnóstico, sistemas para clasificación, sistemas de apoyo a la toma de decisiones.

Ventajas de las Reglas de Producción:

- Naturalidad para representar conocimiento procedimental.
- Modularidad y uniformidad.
- Razonamiento no-monotónico
- Pueden aceptar razonamiento incierto a base factores de certeza.

Desventajas de las Reglas de Producción:

- Posibilidad de encadenamiento infinito.

- Peligro de incorporación de conocimiento nuevo contradictorio.
- Ineficiencia, opacidad (dificultad de establecer relaciones).
- Rápido crecimiento del número de reglas en sistemas de baja a mediana complejidad.

7.5 Razonamiento Inexacto

No siempre es posible contar con toda la información o ésta puede ser incorrecta, incompleta o ambigua. Esto da lugar a diferentes formas de inconsistencia e incertidumbre para las que se han desarrollado modelos de razonamiento inexacto:

- Modelo Aproximado basado en la Teoría de Certeza: *Factores de Incertidumbre*
- Modelo Estadístico - Probabilístico (basado en la Teoría de Bayes): *Valores de Probabilidad*
- Modelo Basado en Lógica Difusa: *Grado de Pertenencia*

7.5.1 Modelo Aproximado Basado en la Teoría de Certeza

A cada regla y a cada componente de una regla se le puede asociar un **factor de incertidumbre**. Este puede ser un número comprendido entre *-1* y *+1*. Para una regla de producción formulada como:

$$\text{IF } E_i \text{ THEN } H_j(c_{ij})$$

La evidencia E_i hace sospechar que la causa (hipótesis) sea H_j. El factor de incertidumbre c_{ij} representa el grado de confianza que el experto que ha propuesto esa regla tiene en ella: supuesto que E_i sea verdadero, $c_{ij} = +1$ correspondería a una seguridad absoluta de que se puede aceptar la hipótesis H_j; $c_{ij} = -1$, a una seguridad absoluta de que se puede descartar H_j; y, $c_{ij} = 0$ correspondería a una incertidumbre o ignorancia absoluta sobre el asunto, lo que haría a la regla totalmente inútil).

7.5.1.1 *Heurísticos para las Deducciones*

Las deducciones inciertas o imprecisas en estos sistemas se hacen con los siguientes heurísticos:

- Si existe la regla IF A THEN $B(c_R)$ y A tiene un factor de incertidumbre c_A, entonces si $c_A \leq 0$ la regla no se aplica, y en caso contrario la regla se dispara y se añade B a la base de hechos, con un factor $c_B = c_A \cdot c_R$.
- En general, A estará compuesto por otros hechos unidos por AND, OR y NOT; el cálculo del factor de incertidumbre resultante se hace de acuerdo con las siguientes fórmulas (derivadas de las **leyes de Lukasiewicz**[119]):

[119] Lukasiewicz, J., *Observaciones filosóficas sobre los sistemas polivalentes de lógica proposicional*. (1930), en Estudios de lógica y filosofía. Trad. de A. Deaño. Rev. Occidente, Madrid. 1975. pp. 61-86.

$c(A1 \text{ AND } A2) = mín(c_{A1}, c_{A2})$

$c(A1 \text{ OR } A2) = máx(c_{A1}, c_{A2})$

$c(\text{NOT } A) = -c(A)$

- Si hay dos reglas que apoyan la misma hipótesis, y una da como resultado un factor c_1 y otra c_2, se aplica para el factor de incertidumbre combinado:

$$c = c_1 + c_2 - c_1.c_2 \quad si \quad c_1.c_2 > 0$$

$$c = \frac{c_1 + c_2}{1 - mín(|c_1|,|c_2|)} \quad si \quad c_1.c_2 < 0$$

Si son más de dos reglas, se calcula el c para las dos primeras, el resultado se combina con el de la tercera, este resultado con el de la cuarta y así sucesivamente.

Para ilustrar cómo se aplican estos heurísticos, supóngase que las reglas sobre enfermedades virales se formulan de manera simple como sigue:

```
R1: IF tiene fiebre THEN padece gripe (0,5)
R2: IF tiene fiebre THEN padece bronquitis (0,1)
R3: IF tiene fiebre THEN padece tuberculosis (0,4)
R4: IF tose mucho THEN padece gripe (0,1)
R5: IF tose mucho THEN padece bronquitis (0,7)
R6: IF tose mucho THEN padece tuberculosis (0,2)
R7: IF tiene dolores musculares THEN padece gripe (0,7)
R8: IF tiene dolores musculares THEN padece bronquitis (0,2)
R9: IF tiene dolores musculares THEN padece tuberculosis (0,1)
R10: IF tiene buen aspecto THEN padece gripe (−0,6)
R11: IF tiene buen aspecto THEN padece bronquitis (−0,2)
R12: IF tiene buen aspecto THEN padece tuberculosis (−1)
```

Si la evidencia en este caso es: bastante seguro que no tiene fiebre, se puede asumir `fiebre(−0,8)`; casi seguro que tiene tos, equivaldría a `tos(0,9);` sin duda tiene dolores musculares, implica `dolores musculares(1);` y, su aspecto es bastante bueno, se interpretaría como `buen aspecto(0,8)`.

Para cada una de las tres hipótesis (`gripe`, `bronquitis`, `tuberculosis`) existen cuatro reglas. Pero las tres primeras no se aplican, porque el antecedente `fiebre` tiene un factor de incertidumbre negativo.

Para la primera hipótesis (gripe) se aplican las reglas R4, R7 y R10:

R4: $0,9 \cdot 0,1 = 0,09$
R7: $1 \cdot 0,7 = 0,7$
R10: $0,8 \cdot (-0,6) = -0,48$

Combinando el resultado de R4 con el de R7, se tiene:

$$0,09 + 0,7 - 0,09 \,.\, 0,7 = 0,727$$

Y combinando éste con el de R10, resulta:

$$c_{gripe} = \frac{0{,}727 - 0{,}48}{1 - 0{,}48} = \mathbf{0,475}$$

Para la segunda hipótesis, ***bronquitis*** se aplican las reglas R5, R8 y R11:

```
R5: 0,9 · 0,7 = 0,63
R8: 1 · 0,2 = 0,2
R11: 0,8 · (-0,2) = -0,16
```

Combinando el resultado de `R5` con el de `R8`, se tiene:

```
0,63 + 0,2 - 0,63 . 0,2 = 0,704
```

Y combinando éste con el de `R11`, resulta:

$$c_{bronquitis} = \frac{0{,}704 - 0{,}16}{1 - 0{,}16} = \mathbf{0,648}$$

Para la tercera hipótesis, ***tuberculosis*** se aplican las reglas R6, R9 y R12:

```
R6: 0,9 · 0,2 = 0,18
R9: 1 · 0,1 = 0,1
R12: 0,8 · (-1) = -0,8
```

Combinando el resultado de `R6` con el de `R9`, se tiene:

```
0,18 + 0,1 - 0,18 . 0,1 = 0,262
```

Y combinando éste con el de `R12`, resulta:

$$c_{tuberculosis} = \frac{0{,}262 - 0{,}8}{1 - 0{,}262} = \mathbf{-0,729}$$

Es decir, la hipótesis de mayor certeza es ***bronquitis***.

7.5.1.2 Sistemas de Unificación de Patrones

Las conclusiones derivadas de las reglas son una colección de hechos que podrían o no unificarse con los varios patrones descritos por las premisas de otras reglas. Sus principales rasgos característicos son:

- Más apropiados para aplicaciones que requieren un proceso de razonamiento progresivo.
- Las relaciones entre las reglas y los hechos son formadas durante la ejecución, basados en los patrones que se identifican con los hechos.
- Los sistemas que utilizan la unificación de patrones son flexibles y poderosos.
- Útiles en dominios en los que las posibles soluciones son, ya sea ilimitadas o muy grandes en número.

Herramientas de desarrollo: PROLOG, XCON, OPS-5, ART, CLIPS, KEE.

7.5.1.3 Redes de Inferencia

Las conclusiones derivadas de las reglas pueden ser hechos que se identifican en forma exacta con las premisas de otras reglas. Sus principales rasgos característicos son:

- Más apropiados para aplicaciones que requieren un proceso de razonamiento regresivo.
- Útiles en dominios donde número de diferentes soluciones alternativas es limitado.
- Fáciles de implementar.
- Se debe conocer de antemano todas las relaciones entre reglas y hechos.

Herramientas de desarrollo: Personal Consultant, EXSYS, VP-Expert.

7.5.2 Redes Semánticas

Es un método de representación gráfico, fácilmente traducible a forma simbólica, muy útil para representar conocimiento taxonómico. Se trata de un lenguaje gráfico, compuesto de nodos y enlaces, en donde los nodos representan objetos y los enlaces las relaciones entre los objetos. Tanto los nodos como los enlaces tienen rótulos que claramente describen los objetos representados y sus relaciones naturales.

Una red semántica permite que sus elementos hereden propiedades a través de los enlaces ES-UN. Para el caso en que un nodo herede información incorrecta, el sistema proporciona la facilidad de manejo de excepciones.

En este modelo se conforman racimos, que serían pequeñas redes de conceptos y propiedades que el procesador cognitivo almacena en o recupera de la memoria a largo plazo en un ciclo de reconocimiento y acción. La siguiente figura muestra el racimo que podría evocarse cuando se piensa en algo asociado a un `animal`.

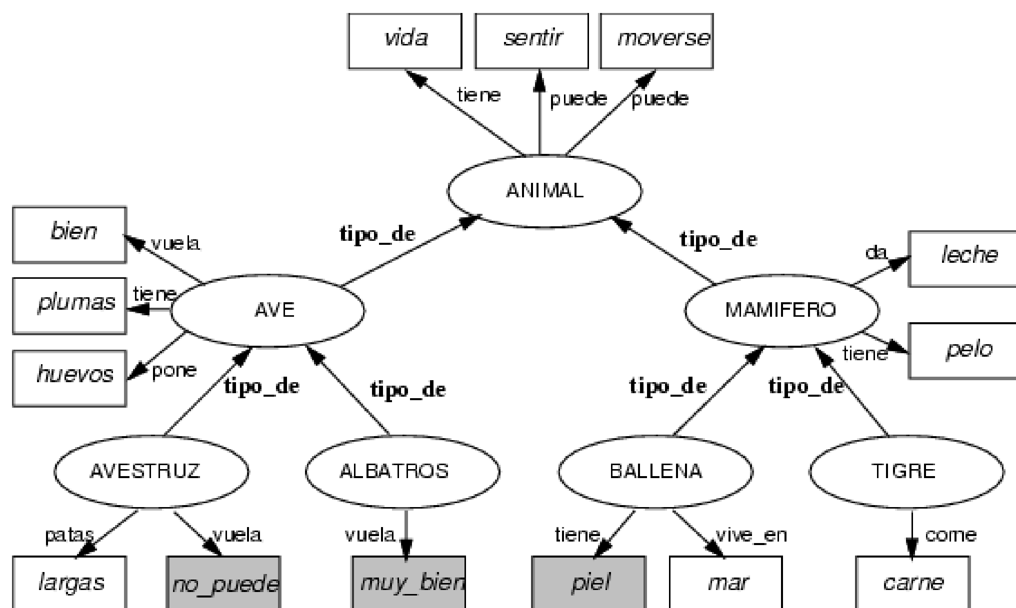


Figura 37. Ejemplo de Red Semántica

Ventajas de las Redes Semánticas

- Permiten declaración de asociaciones en forma explícita y sucinta.
- Procesos de búsqueda de hechos particulares se ejecutan rápidamente.

Desventajas de las Redes Semánticas

- No existe interpretación normalizada para el conocimiento expresado por la red.
- Puede derivar en una explosión combinatoria del número de relaciones que deben ser examinadas para comprobar un hecho en redes complejas.

7.5.3 Marcos (Frames)

Los marcos (*frames*), también conocidos como estructuras o unidades, introducidos por Minsky[120] en 1974, son una forma de expresar las redes semánticas textualmente, pero además pueden incluir representaciones de conocimiento procedimental. En efecto, cada nodo correspondiente a un objeto o a una clase se convierte en un marco, que consta de una primera línea con el nombre del marco y una sucesión de líneas, llamadas ranuras (*slots*) con la sintaxis:

```
<ranura> ::= <nombre de relación>: <objeto relacionado> |
                <nombre de relación>: <clase relacionada> |
                <nombre de propiedad>: <valor de la propiedad> |
                <nombre de propiedad>: (excep) <valor de la propiedad> |
                <nombre de propiedad>: if_needed <procedimiento> |
                <nombre de propiedad>: if_added <procedimiento>
<nombre de relación> ::= es_un |
                         tipo_de |
                         <relacion específica de la aplicación>
```

Las dos últimas líneas de la definición de <ranura> son las que confieren a los sistemas basados en marcos esa posibilidad de mezclar conocimiento procedimental con el declarativo: <if_needed> significa que si se necesita el valor de la propiedad se active un procedimiento para calcularla; <if_added>, que si en el curso del proceso se rellena o se modifica ese valor se active un procedimiento.

A esta adición de procedimientos, llamados también *facetas o demons* a un marco se le llama adosamiento procedimental (*procedural attachment*). Adicionalmente, pueden existir otros procedimientos para controlar los valores de los atributos:

- Limitar valores dentro de un rango o conjunto.
- Restringir los tipos de datos a ser almacenados.
- El valor a ser asumido si no se especifica alguno (default).
- Qué hacer si un valor cambia (if_changed) ...

---

[120] Minsky, M.: *A Framework for Representing Knowledge* [en línea]. Memo 306, MIT AI Lab, June 1974. Disponible en **http://web.media.mit.edu/~minsky/papers/Frames/frames.html**

También es fácil introducir en la representación meta conocimiento, es decir, conocimiento sobre el marco mismo. Como en las tres primeras ranuras de este ejemplo:

```
marco_avestruz
     descripcion:  caracterización  básica  de  un  avestruz
         autor                           :     Prof.    Gallo
          fecha                              :     31/12/2011
              tipo_de                           :        ave
           patas                                :      largas
  vuela      : (exc) no_puede
```

Los marcos pueden ser organizados en estructuras jerárquicas que representan a un concepto u objeto en diferentes niveles de abstracción.

- Ventajas de los marcos (*Frames*)
  - Facilidad de ejecutar procesos guiados por las expectativas.
  - Conocimiento mejor estructurado y organizado que en una red asociativa.
  - Pueden ser capaces de determinar su propia aplicabilidad en ciertas situaciones.
  - Posibilidad de almacenar dinámicamente valores de variables en las ranuras.
- Desventajas de los marcos (*Frames*)
  - Dificultad de representar objetos o conceptos que se alejen de estereotipos.
  - Dificultad de describir conocimiento heurístico.

7.5.4 Representación Mediante Objetos

Un *objeto* es definido como una colección de información y conocimiento representando una entidad del mundo real, incluyendo una descripción de cómo manipular la información o el conocimiento (*métodos*).

Un *objeto* tiene un nombre, una caracterización de clase, varios atributos distintivos y un conjunto de operaciones (*métodos*). Soporta herencia, encapsulamiento, polimorfismo, y otros atributos de la representación orientada a objetos.

La relación entre los objetos viene definida por los *mensajes*. Cuando un objeto recibe un mensaje válido, responde con una acción apropiada.

Ventajas de la Representación Orientada a Objetos:

- Poder de abstracción.
- Encapsulamiento, herencia, polimorfismo.
- Reutilización de código.
- Facilidad de trabajo y eficiencia con grandes sistemas.

Desventajas de la Representación Orientada a Objetos:

- Dificultad de manejar objetos que se alejen demasiado de la norma.

7.6 Sistemas Expertos

Un Sistema Experto es un sistema basado en conocimiento cuya operación genera resultados que emulan lo que se obtendría de un experto humano. Sus características son:

- Solidez en el dominio de su conocimiento.
- Capacidad para resolver problemas.
- Fiabilidad para poder confiar en sus resultados.
- Habilidad para adquirir conocimiento.

Estas características se reflejan en su arquitectura. A los elementos propios de un sistema basado en conocimiento se ha incorporado un módulo de explicación y un módulo de aprendizaje.

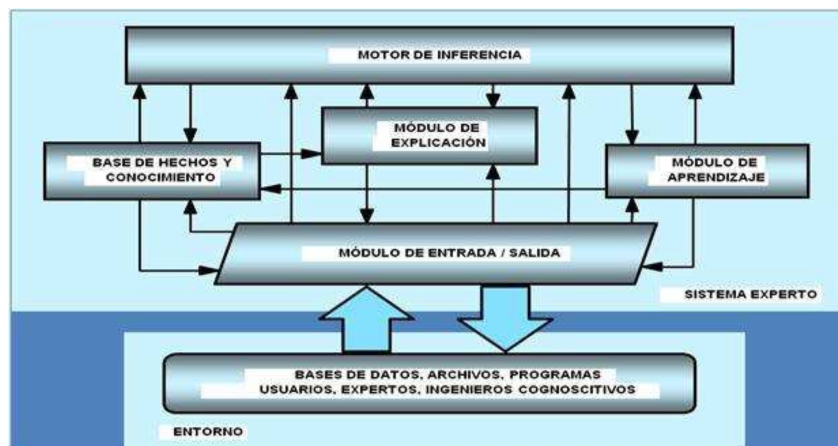


Figura 38. Arquitectura de los Sistemas Expertos

7.7 Tareas Expertas

Entre las tareas que un sistema experto es capaz de realizar, se pueden indicar las siguientes, en orden de menor complejidad a mayor complejidad:

- Interpretación, Análisis y Síntesis
- Diagnóstico
- Supervisión
- Reparación, Corrección, Terapia
- Control: Tiempo Real o Diferido
- Simulación, Pronóstico, Predicción
- Planificación
- Diseño
- Instrucción, Capacitación, Enseñanza, Tutoría

7.8 Ingeniería del Conocimiento

Es una disciplina que trata la integración del conocimiento en sistemas computarizados para resolver problemas complejos que generalmente requieren de un alto nivel de experticia humana[121].

Actualmente, se refiere al diseño, desarrollo, construcción y mantenimiento de sistemas basados en conocimiento. Está relacionada con la lógica matemática, la ciencia cognitiva y la ingeniería de software. Se aplica en varios dominios de las ciencias de computación, tales como inteligencia artificial, bases de datos, minería de datos, sistemas de soporte a decisiones y sistemas de información geográficos. Los principales objetivos de la ingeniería del conocimiento son: analizar, desarrollar y mantener sistemas basados en conocimiento (KBS). Los principios que guían la Ingeniería del Conocimiento se indican a continuación:

- Los métodos deben ser escogidos apropiadamente, según los tipos de expertos y experticia.
- Existen diferentes tipos de conocimiento. Para cada uno de ellos, se debe utilizar el enfoque y la técnica apropiada, según corresponda.
- Existen diferentes formas de representar conocimiento, lo que puede ayudar la adquisición, validación y reutilización del conocimiento.
- Existen diferentes maneras de utilizar el conocimiento, por lo que el proceso de adquisición, debe ser orientado por la metas.
- Se pueden utilizar métodos estructurados, para incrementar la eficiencia del proceso de desarrollo.

7.8.1 Análisis Cognitivo

El análisis cognitivo es la esencia de la ingeniería del conocimiento. El desarrollo de un sistema basado en conocimiento requiere de la adquisición de conocimiento experto. Para esto, se deben identificar fuentes de conocimiento y personas expertas a las que se debe observar o entrevistar para determinar lo que conoce, cómo establece las relaciones entre los diferentes hechos y cómo los integra en un proceso de razonamiento para solucionar problemas en el campo de su dominio. Todos los procesos interesantes que realiza un experto, ocurren en su interior. Externamente, sólo se observa su comportamiento. El experto piensa, razona, formula hipótesis, consulta y realiza preguntas para confirmarlas o rechazarlas.

El análisis cognitivo es un proceso mediante el cual el ingeniero cognitivo descifra lo que un experto sabe y cómo el experto razona acerca de tareas específicas para resolver problemas en el campo de su dominio.

[121] Feigenbaum, Edward A.; McCorduck, Pamela. *The fifth generation*. Reading, MA: Addison-Wesley, USA, 1983.

Uno de los asuntos más estudiados y debatidos en la psicología cognitiva son los referentes a la memoria humana. Es comúnmente admitido que hay tres tipos básicos de memoria a largo plazo:

- La **memoria episódica** guarda recuerdo de eventos (episodios) experimentados personalmente, por ejemplo: ayer tuve una reunión, dónde estuve el lunes, cómo fui a Loja el año pasado ...
- La **memoria semántica** guarda vocabulario, hechos generales, conceptos y relaciones, sin referencia a cómo, dónde o cuándo se han adquirido esos conocimientos: las reuniones largas cansan, la contaminación del aire enferma, para viajes largos es mejor el avión.
- La **memoria procedimental** guarda patrones de actuación frente a eventos o problemas específicos, patrones que pueden ser elementales (planificar reuniones cortas, evitar horas pico en el centro histórico) o compuestos (secuencia de pasos para hacer una reserva de vuelo por Internet).

La memoria semántica fue el primer modelo estructural de memoria[122], y ha tenido mucha influencia en la ingeniería del conocimiento y en la ingeniería del software en general. La idea esencial es que la mente llega a formar conceptos agrupando elementos de la memoria episódica con el fin de conseguir una economía (formación de racimos).

Precisamente, el análisis cognitivo está basado en la investigación científica acerca de los niveles de conocimiento y la estructura jerárquica de la memoria humana.

En cuanto a la forma de clasificar el conocimiento, usualmente se identifican tres categorías:

- Conocimiento superficial o heurístico (Hechos).
- Conocimiento de dominio (Modelos procedimentales, relacionales, algoritmos).
- Conocimiento teórico o profundo (Teorías abstractas, leyes y principios).

Por otra parte, experimentos realizados por Collins y Quillian[123] evidenciaron que en la mente de los humanos los conceptos se almacenan en forma jerárquica, asociándose entre sí directa o indirectamente, y que para recuperar las propiedades asociadas a tales conceptos el procesador cognitivo tiene que recorrer las asociaciones.

Un ingeniero cognitivo dispone de tres grupos de herramientas para analizar tareas cognitivas:

---

[122] Quillian, M.R. Semantic memory. En M. Minsky (ed.): *Semantic Information Processing*. MIT Press, Cambridge, Mass., 1968, pp. 354–402.

[123] Collins, A. M. Y Quillian, M.R.: *Retrieval Time from Semantic Memory*. Journal of Verbal Learning and Verbal Behaviour, 8 (1969), pp 240–247.

1. **Herramientas Intelectuales**.- En la organización jerárquica del conocimiento, el análisis cognitivo trata de identificar agrupaciones de objetos abstractos que constituyen el conocimiento del experto, para desarrollar modelos de dominio. Se entiende por dominio a un conjunto circunscrito de actividades. Ejemplos de dominios son: el futbol, la contabilidad, tocar guitarra, la automatización industrial, las finanzas, etc. El ingeniero cognitivo desarrolla modelos de dominio asociando a modelos análogos que le son conocidos, para obtener una visión general de la estructura del modelo que el experto podría estar utilizando.
2. **Herramientas de Anotación**.- El ingeniero cognitivo tiene un vocabulario básico que le permite describir el conocimiento del experto en términos de objetos, atributos, valores y relaciones. Además, utiliza convenciones para representar jerárquicamente objetos de interés y para configurar modelos procedimentales.
3. **Herramientas Empíricas**.- Las dos herramientas anteriores son limitadas, sólo ayudan a iniciar el proceso. Una vez que se haya desarrollado un modelo del dominio y el conjunto de reglas para describir el conocimiento del experto, el ingeniero cognitivo debe desarrollar algún método empírico para refinar el conocimiento adquirido durante el análisis cognitivo. Este conocimiento debe ser codificado e integrado en un sistema computarizado (prototipo), al que se somete a pruebas y refinamiento, hasta que el sistema logre aproximar aceptablemente el desempeño del experto humano.

7.8.2 Adquisición del Conocimiento

Este es un proceso complejo que, en el sentido más específico, permite extraer, simplificar, modelar y traducir el saber de diversos dominios, en un conjunto preciso y fácilmente modificable de hechos y reglas que configuran un sistema basado en conocimiento.

En un sentido más amplio, involucra todo el proceso de ingeniería del conocimiento, incluyendo la entrevista a los expertos, el desarrollo de un mapa de conocimientos, la codificación del conocimiento, el diseño, construcción, pruebas y afinamiento del sistema, interactuando con expertos y usuarios, hasta que esté listo para su entrega, instalación e implantación.

A pesar de que no es posible observar directamente el conocimiento de alguien, sí se puede observar y aprender a identificar la experticia. Se distinguen dos tipos de experticia:

- Motora o basada en habilidad.- Operar o controlar un equipo.
- Cognoscitiva.- Realizar un diagnóstico o prescribir un tratamiento.

En cualquiera de los dos casos, la experticia no es sino una demostración de la forma en que se aplica el conocimiento. A pesar de que dos individuos posean similar conocimiento, no se puede predecir que ambos demostrarán igual uso del conocimiento. De hecho, habrá diferencia. Precisamente esta diferencia es la noción de experticia.

En el marco conceptual para la adquisición de conocimiento, pueden distinguirse varias actividades interrelacionadas tal como se indica en la siguiente figura.

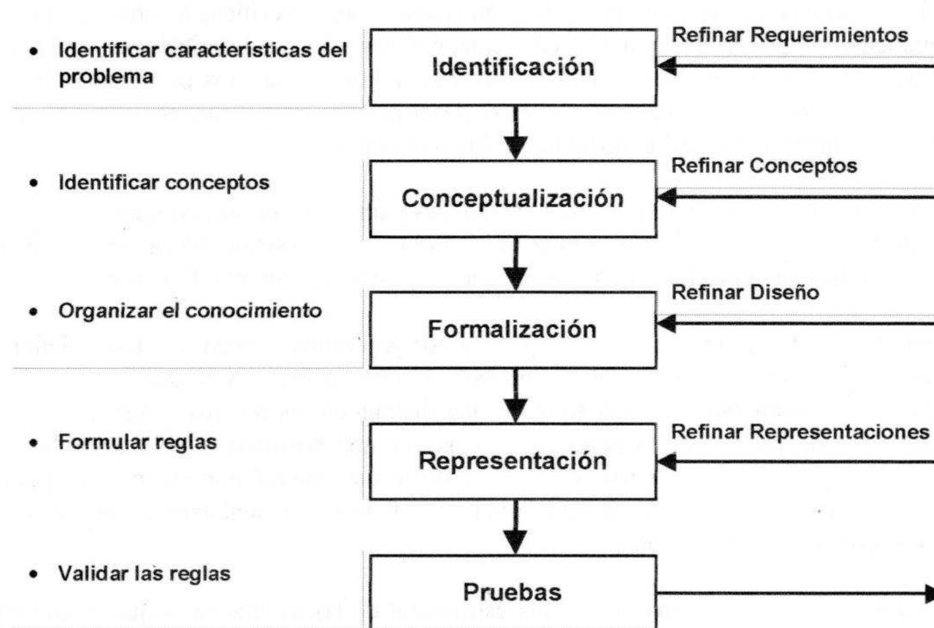


Figura 39. Etapas de la Adquisición del Conocimiento

- **Identificación**.- Corresponde al proceso de caracterizar los aspectos claves del problema, incluyendo participantes, rasgos, atributos, recursos y metas. Durante esta etapa, el ingeniero de conocimiento se familiariza con el dominio, establece los objetivos iniciales para el desarrollo y selecciona expertos en el dominio y otras fuentes apropiadas que están disponibles.
- **Conceptualización**.- Trata de especificar la forma en que los conceptos primarios y relaciones claves entre los conceptos dentro del dominio, son presentados y son interrelacionados por los expertos. Este es un paso muy difícil, que conceptualmente podría dar lugar a la proyección de la forma en que el conocimiento asumido está organizado.
- **Formalización**.- En esta etapa el ingeniero de conocimiento debe proyectar los conceptos reconocidos, las tareas, relaciones y otra información disponible, en mecanismos formales de representación. La identificación de un modelo fundamental que los expertos utilizan para generar soluciones es un importante logro en esta etapa.
- **Representación**.- Una vez que el conocimiento ha sido formalizado, se lo puede representar. Las tareas primordiales del ingeniero de conocimiento en esta etapa son: evaluar el esquema de representación seleccionado; y, preparar la documentación detallada que permita enlazar el contenido de la base de conocimiento con sus fuentes originales.
- **Pruebas**.- La etapa de prueba permite evaluar la eficacia de las actividades realizadas en las etapas anteriores. Los resultados pueden usarse para refinar los conceptos iniciales refinar los esquemas de representación del conocimiento, sus interrelaciones; y, modificar o revisar los métodos de adquisición del conocimiento.

Se pueden distinguir cuatro técnicas fundamentales para la adquisición del conocimiento[124]:

- **Entrevista**.- Son de tipo verbal en las que el experto, fuera de su ambiente de trabajo, es requerido que reflexione, recuerde y explique su metodología de resolución de problemas. La entrevista puede ser de varios tipos: focalizada, estructurada, introspectiva, y tutorial.
- **Observación**.- En este caso el experto se encuentra en una situación real o simulada de resolución de problemas. Los diferentes tipos de técnicas de observación son: auto reportaje, observación participativa, y diálogo usuario - experto.
- **Técnicas multidimensionales**.- Normalmente producen datos no verbales. Usualmente se fuerza al experto a pensar en el dominio de sus conocimientos en una forma nueva, a tal punto que parecería que no está relacionada con las tareas que efectivamente realiza. Las posibles variantes pueden ser: grilla de repertorio, ordenamiento de conceptos y, generación matricial.
- **Revisión**.- Es una técnica general, suplementaria, pero importante y necesaria, para garantizar la efectividad de las técnicas descritas anteriormente.

La siguiente tabla presenta sugerencias para relacionar el tipo de conocimiento con técnicas para su adquisición.

TIPO DE CONOCIMIENTO	ACTIVIDAD	TÉCNICA SUGERIDA
Declarativo	Identificación de heurística general (consciente).	Entrevistas
Procedimental	Identificación de tareas y procedimientos rutinarios.	Entrevista estructurada Observación de procesos Simulaciones
Semántico	Identificación de vocabulario y conceptos	Cuadrícula de repertorio[125] Ordenamiento de conceptos
	Identificación de procedimientos de toma de decisiones y heurísticos (inconsciente).	Análisis de tareas Observación de procesos
Episódico	Identificación de heurísticas para resolución de problemas por analogía.	Simulaciones Observación de procesos

Figura 40. Correlación entre tipo de conocimiento y técnicas de adquisición.

124 Karen L. McGraw and Karan Harbison Briggs. *Knowledge Acquisition: Principles and Guidelines*. Prentice Hall International Editions, USA, 1989.

125 http://edutechwiki.unige.ch/en/Repertory_grid_technique

Una metodología para adquisición del conocimiento que incluye elementos de ingeniería de sistemas e ingeniería de software se presenta a continuación. Representa una adaptación del modelo de cascada y permite un rápido prototipaje, debidamente administrado.

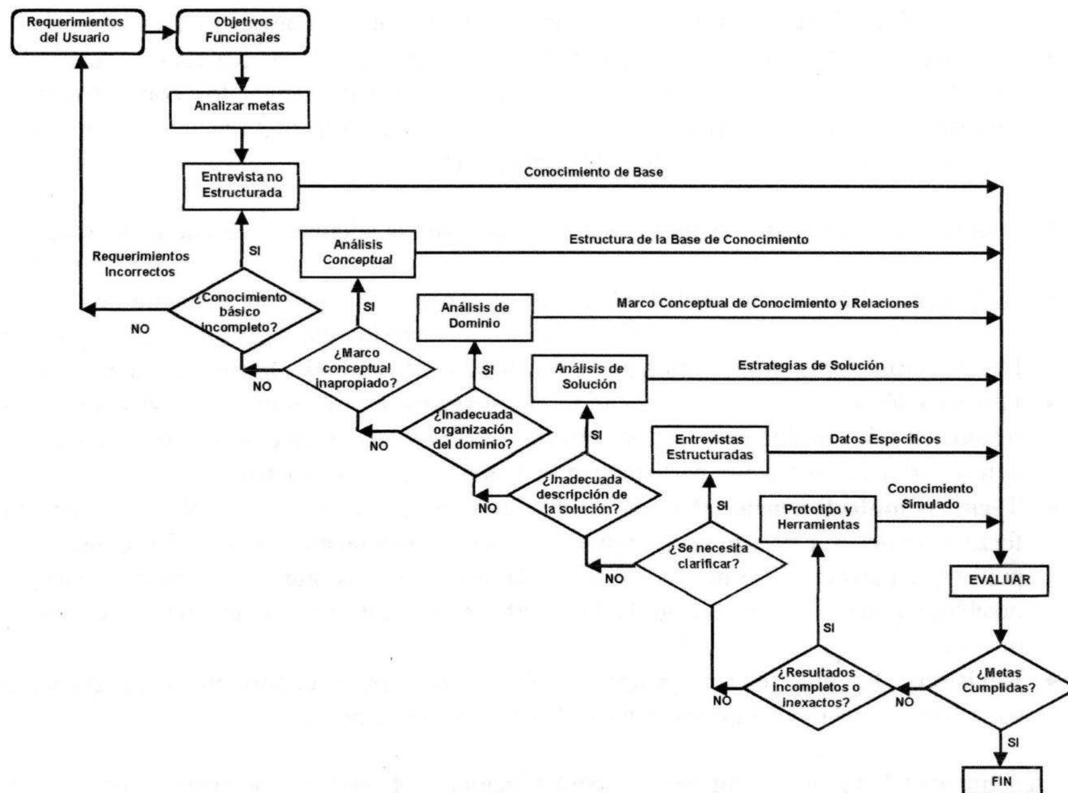


Figura 41. Proceso adaptativo en cascada para adquisición del conocimiento

7.9 Ingeniería de Sistemas Basados en Conocimiento

Para el desarrollo de un sistema basado en conocimiento no existen metodologías o marcos de trabajo (*frameworks*) estandarizados como para el caso de ingeniería de software. Sin embargo, existen algunas propuestas que se han configurado, para apoyar su desarrollo:

- RAPID
- Knowledge-based Analysis and Design System (KADS)
- CommonKADS

7.9.1 RAPID

Al ser el conocimiento un insumo ambiguo, difuso y estocástico, para el desarrollo de sistemas basados en conocimiento se recomienda seguir un proceso denominado **RAPID,** por los nombres en inglés de sus dos fases: ***Rap****id Prototyping and* ***I****ncremental* ***D****evelopment* (Prototipaje rápido y desarrollo incremental). En ambas fases del desarrollo del sistema, los factores clave del éxito del proyecto son el involucramiento y la interacción de los principales agentes: desarrollador o ingeniero cognitivo (***knowledge engineer***), usuario y expertos.

Análisis.- Con el apoyo de expertos en el campo del conocimiento específico, evaluar el problema y los recursos disponibles para determinar la aplicabilidad de una solución basada en conocimiento, incluyendo un análisis beneficio - costo.

Especificación de Requerimientos.- Con la intervención de usuarios y expertos, el desarrollador debe formalizar y poner por escrito lo que fue adquirido durante la fase de análisis. Esto permite determinar los objetivos del proyecto, la funcionalidad esperada, los atributos de calidad, establecer los medios para alcanzarlos y registrar legalmente los compromisos establecidos de común acuerdo con los usuarios del sistema.

Diseño Preliminar.- El desarrollador con el asesoramiento de los expertos, considera únicamente las decisiones de alto nivel necesarias para preparar y desarrollar rápidamente el prototipo inicial. En esta etapa se determina el paradigma de representación del conocimiento y se escoge la herramienta que servirá para construir el prototipo inicial.

Prototipo Inicial y Evaluación.- Sobre la base del conocimiento recopilado de los expertos y la información registrada de los usuarios en las etapas anteriores, el desarrollador construye un prototipo inicial que, con la ayuda de los usuarios y expertos, le permitirán confirmar, rectificar o desechar las decisiones tomadas en el diseño preliminar.

Diseño Inicial.- Con las experiencias obtenidas de la evaluación del prototipo inicial, el desarrollador realiza un diseño preliminar del sistema con las características y funcionalidades esperadas en la versión de entrega, por lo que se debe también incluir el diseño de un **plan de pruebas**. Con esta etapa termina la fase de **prototipaje rápido**.

Implementación.- Con esta etapa se inicia la fase de **desarrollo incremental**. Con la participación de usuarios y expertos, el desarrollador construye todos los módulos del sistema, utilizando herramientas que pueden o no ser las mismas que fueron utilizadas para el prototipo inicial.

Pruebas.- La aplicación del plan de pruebas permite verificar la operación del sistema y todas sus unidades, así como validar los resultados obtenidos al aplicar casos de prueba certificados por los expertos.

Ajustes al diseño.- Los resultados obtenidos son sometidos a la evaluación de usuarios y expertos, a fin de identificar posibles ajustes al diseño que, de ser el caso, son implementados y sometidos a posteriores pruebas, hasta obtener la aceptación de los usuarios, basada en la satisfacción de requerimientos y criterios de calidad inicialmente convenidos.

Instalación, Implantación y Mantenimiento.- Terminada la etapa de desarrollo incremental, se procede a la instalación del sistema. Luego viene el proceso de puesta en producción (implantación) y la conformación de un plan de mantenimiento y actualización del sistema.

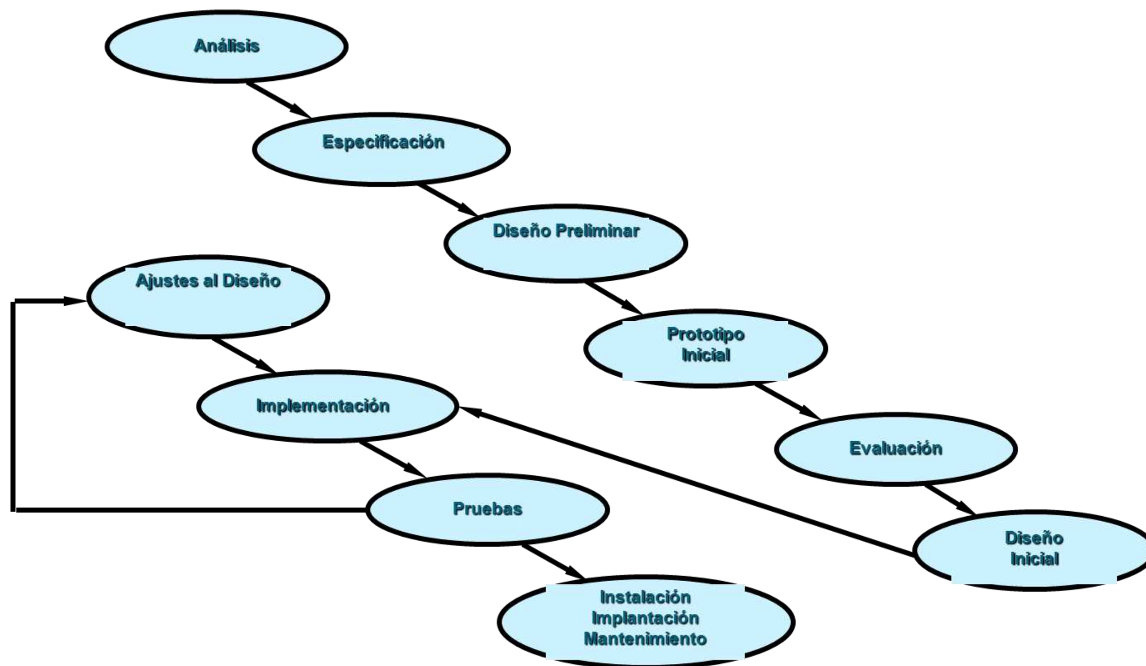


Figura 42. Proceso de prototipaje rápido y desarrollo incremental (RAPID)

7.9.2 Knowledge-based Analysis and Design System (KADS)[126]

Los métodos propuestos por KADS representan aproximaciones estructuradas y sistemáticas a dos áreas clave del desarrollo de sistemas basados en conocimiento: las fases de análisis y de diseño. En general, KADS se compone de:

- Actividades estructuradas para análisis de requerimientos y diseño hasta el nivel de módulos de un sistema basado en conocimiento.
- Definiciones de resultados o entregables, producidos por las actividades de análisis y diseño.
- Asesoramiento y guía en las técnicas a utilizarse para realizar las actividades de análisis y diseño.
- Asesoramiento y guía en las herramientas, software, incluyendo bibliotecas que pueden ser usadas para apoyar a dichas técnicas.
- Apoyo para el control y aseguramiento de la calidad en la producción de resultados.
- Identificación de oportunidades para producción de prototipos.

---

[126] Tansley, D S W and Hayball C C. *Knowledge-Based Systems Analysis and Design: A KADS Developer's Handbook*. The BCS Practitioner Series. Prentice Hall, UK, 1993.

7.9.3 CommonKADS[127,128]

En Europa, se considera un estándar de facto para análisis de conocimiento y desarrollo de sistemas basados en conocimiento. Es una mejora del KADS. Describe el desarrollo de un sistema basado en conocimiento desde dos perspectivas:

- **Perspectiva de resultado**.- Un conjunto de modelos de diferentes aspectos de un sistema basado en conocimiento y su entorno, los cuales se mejoran continuamente, durante el ciclo de vida del proyecto.
- **Perspectiva de gestión de proyecto**.- Un modelo de ciclo de vida en espiral, guiado por el riesgo, que puede ser configurado como un proceso adaptado a un proyecto en particular.

7.10 Herramientas de Desarrollo

El éxito de la aplicación de un sistema basado en conocimiento, en términos de funcionamiento, usabilidad, mantenimiento y portabilidad, depende fundamentalmente de las herramientas de desarrollo e instalación utilizadas. Consecuentemente, la selección de la herramienta de programación o desarrollo del sistema es una tarea clave en la planificación del proyecto.

Idealmente las herramientas escogidas deberían ser independientes del hardware y software de los entornos de trabajo, es decir, capaces de ejecutarse en cualquier plataforma. Adicionalmente, se incorporan consideraciones de: costo – beneficio de la herramienta, curva de aprendizaje, tiempo y facilidades para el desarrollo del sistema y versatilidad en el soporte de paradigmas de representación del conocimiento, entre otras. El espectro de herramientas para desarrollo de sistemas basados en conocimiento es amplio, variado y permanentemente cambiante, va desde el uso de lenguajes de programación, hasta entornos de desarrollo. La siguiente es una muestra:

- Lenguajes de Programación
  - Imperativos: Pascal, C/C++, ...
  - Funcionales: Lisp, ...
  - Declarativos: PROLOG, ...
  - Orientados a Objetos: SmallTalk, CLOS, ...
  - Orientados a la WEB: Java, …
- Entornos de Desarrollo
  - Sistemas Vacíos (*Shells*): ExSys, VP-Expert, Crystal, ...
  - Entornos Híbridos: CLIPS, Nexpert Object, ART, ...

---

[127] **http://www.sics.se/ktm/kads.html**

[128] Schreiber, August Th.; Akkermans, Hans; Anjewierden, Anjo; Dehoog, Robert; Shadbolt, Nigel; Vandevelde, Walter; Wielinga, Bob (2000), *Knowledge engineering and management: the CommonKADS methodology*. The MIT Press, USA, 1999.

7.11 El Entorno de Desarrollo CLIPS

El ***C L****anguage* ***I****ntegrated* ***P****roduction* ***S****ystem* (**CLIPS**)[129] se originó en el *Software Technology Branch* del Centro Espacial *Lindon B. Johnson* de la *NASA*. Desde la aparición de su primera versión en 1986, CLIPS ha estado en constante mejoramiento. Actualmente se lo mantiene independiente de la NASA, como software de dominio público. CLIPS es una herramienta productiva que proporciona un entorno completo para el desarrollo de sistemas expertos cuyo conocimiento puede ser representado por:

- Reglas de producción.- Apropiadas para conocimiento heurístico o basado en experiencia.
- Funciones genéricas y funciones definidas por el usuario.- Apropiadas para conocimiento procedimental.
- Programación orientada a objetos.- Apropiadas también para conocimiento procedimental. La implementación soporta 6 características generalmente aceptadas en los sistemas orientados a objetos, estas son: clases, abstracción, encapsulamiento, herencia, polimorfismo y gestión de mensajes.

Se pueden desarrollar aplicaciones con cualquier combinación de estas representaciones. Igualmente, CLIPS permite la integración con otros lenguajes como el C.

A CLIPS se lo considera una herramienta híbrida ya que a más de las construcciones procedimentales típicas de un lenguaje de programación, es un entorno de desarrollo que incluye a más del editor y de la herramienta de depuración, un motor de inferencia. En consecuencia, CLIPS proporciona los elementos básicos que se requieren para la construcción de un sistema experto:

- Una memoria global para datos o hechos: `fact-list` e `instance-list`.
- Una base de conocimientos y una base de reglas: `knowledge-base`, `rule-base`.
- Un motor de inferencia que controla la ejecución de todas las reglas.

Un programa en CLIPS puede consistir de hechos, reglas y objetos. El motor de inferencia decide qué reglas deberían ejecutarse y cuándo. Desde esta perspectiva, un sistema experto escrito en CLIPS, es un programa guiado por los datos (hechos y objetos) que estimulan la ejecución a través del motor de inferencia.

Para trabajar con CLIPS, se ejecuta el comando `CLIPSWin.exe`. Con esta acción, aparece el entorno de desarrollo, con su ventana de diálogo y la barra de comandos. A ésta se la puede agregar 5 ventanas de monitoreo, como se aprecia en la siguiente figura:

---

[129] http://clipsrules.sourceforge.net/index.html

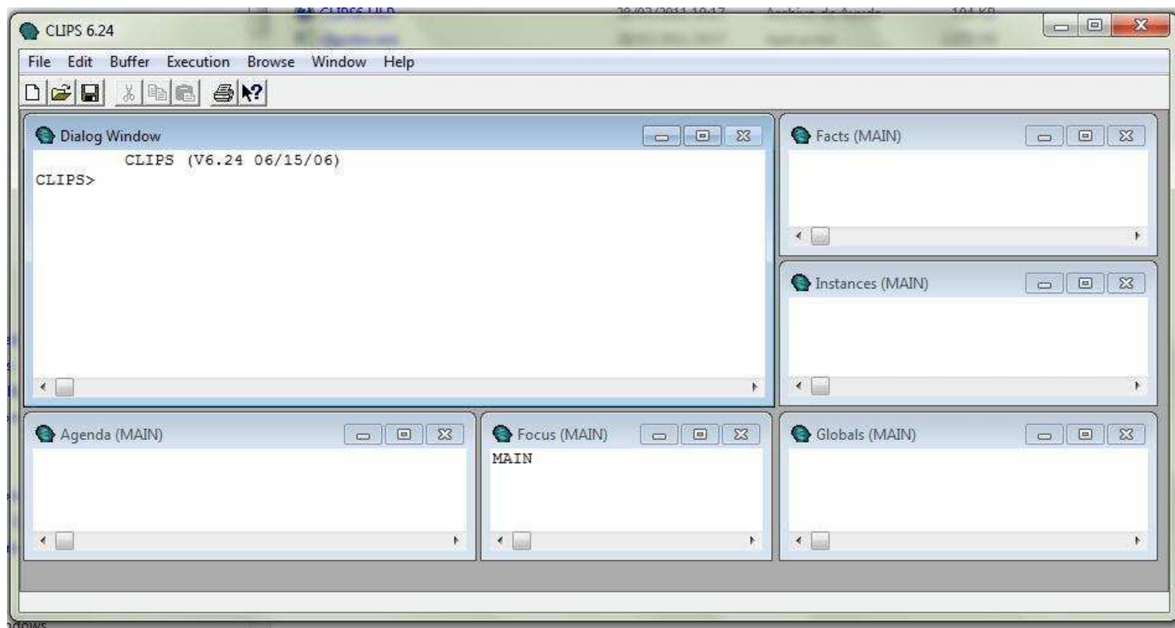


Figura 43. Entorno de trabajo de CLIPS

En la ventana de diálogo aparece la notificación del sistema `CLIPS>` indicando que está listo para recibir comandos por parte del usuario. Se puede ingresar código directamente desde la notificación del sistema o hacer que el sistema lea e interprete desde un archivo de texto (`File->Load`). En este último caso, el archivo puede ser abierto por el editor del sistema, utilizando `File-> Open`.

7.11.1 Definiciones en CLIPS

Las representaciones de objetos en CLIPS se las hacen por medio de **campos**. Un campo es un contenedor que puede tener un valor asociado. Para indicar que un campo existe pero no tiene valor asociado, se utiliza el símbolo especial **nil**.

CLIPS dispone de diferentes tipos de campos: **`float`**, **`integer`**, **`symbol`**, **`string`**, **`external-address`**, **`fact-address`**, **`instance-name`** e **`instance-address`**. El tipo de cada campo es determinado por el tipo de valor que se guarda en él.

El tipo de campo **`symbol`** tiene un nombre que inicia con un caracter **`ASCII`** imprimible, seguido de cero o más caracteres imprimibles.

Un **`string`** está delimitado por dobles comillas. Las comillas son parte del campo. Dentro de las comillas se pueden poner cero o más caracteres.

Los campos numéricos pueden ser de tipo **`float`** o **`integer`**. En CLIPS, todos los números son tratados como de **`double-precision`** o **`long integer`**.

Múltiples campos se separan por espacios en blanco (espacios, tabulaciones, cambios de línea). Cuando uno o más campos están delimitados o rodeados por paréntesis, toman el nombre de **`facts`** (hechos). Los siguientes son ejemplos de hechos:

(colores amarillo azul rojo)	**(telefono-domicilio 6008000)**
(sueldo_basico 375.78)	**(fallas-por-lote nil)**
(Juan estudiante ing-sistemas)	**(felinos gato tigre pantera león)**
(menu-del-dia "sopa de pollo")	**(distancias 3.5e3 5.7E5)**
(mas-vale-solo) **("amigo")** **(amigo)** **(Amigo)**	**(lista_de_compras arroz** **azucar** **patatas** **leche)**

Como se indica en los ejemplos es recomendable utilizar el primer campo de un hecho, para describir la relación de los siguientes campos.

Cabe señalar que CLIPS hace diferencia entre letras mayúsculas y minúsculas. Por lo que (`amigo`) y (`Amigo`) son hechos diferentes. Además no permite que los hechos sean anidados, es decir un hecho no puede contener a otro hecho. Lo siguiente es ilegal:

```
(mis-mascotas (roedor hamster)) (felino (gato)) ; Esto es ilegal!!!
```

El caracter punto y coma (**;**) se utiliza para iniciar un **comentario**, hasta el final de la línea.

Al igual que cualquier lenguaje de programación, CLIPS tiene **variables** para almacenar valores. El identificador de la variable siempre se escribe iniciando con un signo de interrogación, seguido un símbolo, que es el nombre de la variable. El formato general es: **?<nombre-de-variable>**

```
?color  ?direccion   ?x  ?respuesta  ?Nombre
```

Todas las variables en CLIPS son locales, sin embargo se las puede definir también como globales (**defglobal**).

Antes de que una variable pueda ser referida o utilizada, debe tener asignado un valor. Esto se puede hacer por unificación en el antecedente de una regla, con el comando **bind** y con el operador de asignación de direcciones de objetos "**<-**".

El signo de interrogación (**?**)**,** por sí solo, se utiliza como un ***single-field wildcard***, es decir representa **exactamente un campo**. En cambio, el signo de dólar seguido por el de interrogación **$?** representa un ***multifield wildcard.*** Ejemplos:

Hechos registrados	
(persona Mary Padilla)	(persona Juan Padilla)
(persona Mary Amores)	(persona Luis Eduardo Duque)
(persona Juan Pedro Amores)	(persona Luis Villa)

Patrones	**Interpretación**
(persona ? Padilla)	Personas con dos nombres y el segundo sea Padilla
(persona Luis ? ?)	Personas con tres nombres registrados y el primero sea Luis
(persona Mary ?)	Personas con dos nombres y el primero sea Mary
(persona Luis $?)	Personas cuyo primer nombre sea Luis
(persona $? Amores)	Personas cuyo último nombre sea Amores

7.11.2 Comandos Básicos

A continuación se tabulan algunos comandos básicos que dispone CLIPS:

`assert`	Registra un hecho en la memoria	`watch ...`	Permite observar hechos, reglas, etc.
`facts`	Muestra los hechos registrados	`unwatch ...`	Apaga la observación de hechos, ...
`clear`	Limpia la memoria del sistema	`reset`	Inicializa el sistema
`run`	Ejecuta un programa	`exit`	Sale del entorno y cierra CLIPS
`bind`	Asigna un valor a una variable	`printout t`	Imprime en la salida estándar
`retract`	Retira un hecho registrado previamente		

7.11.3 Definición de Reglas

Las reglas en CLIPS siguen la siguiente sintaxis:

```
(defrule <nombre_regla> "comentario opcional"
      (condición_1) ; Lado izquierdo (LHS)
      (condición_2) ; de la regla consiste de condiciones
                  . ; antes del signo "=>"
                 ..
      (condición_N)
=>                ; Signo Entonces (THEN)
      (acción_1)  ; Lado derecho (RHS)
      (acción_2)  ; de la regal consiste de acciones
                  . ; si se cumplen las condiciones antes de "=>"
                  .
      (acción_M)) ; el último ")" balancea el de apertura
                    ; "(" a la izquierda de "defrule".
                    ; Asegúrese que todos los paréntesis estén
                    ; balanceados o aparecerán mensajes de error.
```

En un programa puede haber sólo un nombre de regla. Las condiciones de una regla son **patrones**. Un patrón consiste de uno o más campos. Cada campo del patrón debe ser una **restricción literal**, es decir deben tener relación con **valores constantes**. CLIPS trata de unificar los patrones de las reglas con hechos registrados en la **`fact-list`**. Si todos los patrones de la regla se identifican con los hechos registrados, la regla se activa y se coloca en la **agenda**.

La agenda es una colección de reglas activadas. Puede contener cero o más activaciones. CLIPS ordena las activaciones en la agenda en términos de prioridad (**`salience`**) y determina cual es la activación apropiada a ser disparada. Si no se declara explícitamente la prioridad de una regla (**`declare salience`**), CLIPS asigna el valor de cero (**`default`**). El rango de valores para la prioridad (**`salience`**) va desde -10.000 hasta 10.000. La ejecución para cuando la agenda está vacía. El comando **`(agenda)`** permite listar las activaciones pendientes de ejecución.

La flecha "**=>**" que cierra el bloque de condiciones y precede al bloque de acciones, es el operador de implicación `ENTONCES` (**`THEN`**).

La última parte de la regla es un bloque donde pueden haber cero o más acciones que serán ejecutadas cuando la regla se active. Cada acción es como una función pero que no retorna valor, sin embargo realiza algo útil al ejecutarse.

Cuando se ejecuta el comando (**reset**) se limpia la memoria, se define **initial-fact** e ingresa a la **fact-list** con el identificador "**f-0**". Cuando una regla no tiene condiciones, utiliza **initial-fact**.

7.11.4 Definición de Hechos

Para registrar un conjunto de hechos sin utilizar cada vez el comando **assert** se puede utilizar la palabra clave **deffacts**. Su sintaxis es la siguiente:

```
(deffacts <nombre_hechos> "comentario opcional"
     (fact1)
       ...      ; hechos a ser registrados
     (factN))
```

7.11.5 Definición de Plantillas

Las plantillas son una ayuda especialmente para escribir reglas que tienen una estructura bien definida. La palabra clave **deftemplate** define un grupo de campos relacionados. A cada campo designado se denomina **slot** (ranura). Pueden designarse como **single-slot** al que tiene exactamente un campo y **multislot** al que contiene cero o más campos.

```
(deftemplate <nombre_plantilla> "comentario opcional"
   (slot nombre1
      (type allowed-values)
      (default algún_valor1))
       ...      ; N slots
   (slot nombreN
      (type allowed-values)
      (default algún_valorN)))
```

Los **allowed-values** pueden ser cualquiera de los tipos primitivos **symbol**, **string**, **integer**, **float** u otros.

7.11.6 Definición de Funciones

La definición de funciones en CLIPS se realiza con la siguiente sintaxis:

```
(deffunction <nombre_funcion> "comentario opcional"
   (?arg1 ?arg2 ...?argM [$?argN]) ;lista de argumentos. El último arg
   (<accion1>                      ;puede ser opcionalmente multifield.
    <accion2>       ;acción1 hasta
    ...             ;acción(K-1) no
    <accion(K-1)>   ;retorna valores
    <accionK>)      ;sólo lo hace la última acción.
```

CLIPS tiene también funciones predefinidas, algunas se muestran a continuación:

LÓGICAS		ARITMÉTICAS	
not	Negación booleana	/	División
and	Conjunción booleana	*	Multiplicación
or	Disyunción booleana	+	Suma
		-	Resta

RELACIÓN O COMPARACIÓN	
eq	Igualdad (cualquier tipo). Compara tipo y magnitud
neq	Diferente (cualquier tipo)
=	Igualdad tipo numérico. Compara magnitud
<>	Desigualdad tipo numérico
>=	Mayor que o igual
>	Mayor que
<=	Menor o igual
<	Menor que

FUNCIONES PREDICADO	CHEQUEA SI <arg> ES:
(evenp <arg>)	Número par
(floatp <arg>)	Número flotante
(integerp <arg>)	Número entero
(lexemep <arg>)	Símbolo o string
(numberp <arg>)	Flotante o entero
(oddp <arg>)	Número impar
(pointerp <arg>)	Dirección externa
(secuencep <arg>)	Valor multicampo
(string <arg>)	String
(symbolp <arg>)	Símbolo

7.11.7 Definición de Clases

En la programación orientada a objetos, una clase es una especie de plantilla que describe características comunes o atributos de objetos. Las clases de objetos se organizan en forma jerárquica en un gráfico, para describir las relaciones de los objetos en un sistema.

Cada clase es una abstracción de un sistema de la vida real que se está tratando de modelar. Las otras características de la programación orientada a objetos también son soportadas por CLIPS: herencia, encapsulación, polimorfismo y vinculación dinámica.

El comportamiento dinámico de un objeto está definido por sus **`message-handlers`**. El **`message-handler`** de un objeto responde a mensajes y toma acciones específicas.

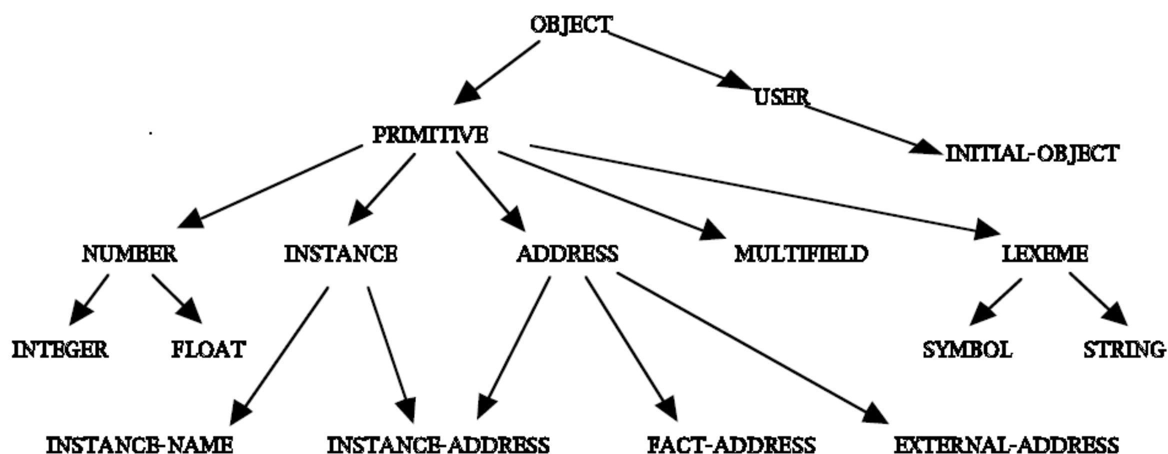


Figura 44. Clases predefinidas en CLIPS

Se recomienda que todas las clases definidas por el usuario sean subclases de USER. De esta forma CLIPS automáticamente proveerá **handlers** para **print**, **init** y **delete**.

El formato básico para definir sólo una clase sin ranuras es:

```
(defclass <clase> (is-a <superclase-directa>))
```

La instancia de una clase se especifica entre corchetes: [objeto]. Un mensaje comienza con la función **send** seguida por el nombre de la instancia, el nombre del mensaje y cualquier argumento requerido: (**send** [**automovil**] **print**). Hace que el **message-handler** imprima los valores de los **slots** de la instancia de la clase **vehículo** llamada **automóvil**.

7.12 Programas de Ejemplo Incluidos en CLIPS

7.12.1 Dilema del Granjero

```
;;;======================================================
;;;   Farmer's Dilemma Problem
;;;     Another classic AI problem (cannibals and the
;;;     missionary) in agricultural terms. The point is
;;;     to get the farmer, the fox the cabbage and the
;;;     goat across a stream.
;;;        But the boat only holds 2 items. If left
;;;     alone with the goat, the fox will eat it. If
;;;     left alone with the cabbage, the goat will eat
;;;     it.
;;;        This example uses rules and object pattern
;;;     matching to solve the problem.
;;;
;;;     CLIPS Version 6.0 Example
;;;
;;;     To execute, merely load, reset and run.
;;;======================================================
;;;***********
;;;* CLASSES *
;;;***********
;;; The status instances hold the state
;;; information of the search tree.
 (defclass status (is-a USER)
   (role concrete)
   (pattern-match reactive)
   (slot search-depth
     (create-accessor write)
     (type INTEGER) (range 1 ?VARIABLE) (default 1))
   (slot parent
     (create-accessor write)
     (type INSTANCE-ADDRESS) (default ?DERIVE))
   (slot farmer-location
     (create-accessor write)
     (type SYMBOL) (allowed-symbols shore-1 shore-2) (default shore-1))
   (slot fox-location
     (create-accessor write)
     (type SYMBOL) (allowed-symbols shore-1 shore-2) (default shore-1))
   (slot goat-location
     (create-accessor write)
     (type SYMBOL) (allowed-symbols shore-1 shore-2) (default shore-1))
   (slot cabbage-location
     (create-accessor write)
     (type SYMBOL) (allowed-symbols shore-1 shore-2) (default shore-1))
   (slot last-move
     (create-accessor write)
     (type SYMBOL) (allowed-symbols no-move alone fox goat cabbage)
     (default no-move)))
;;; The moves instances hold the information of all the moves
;;; made to reach a given state.
 (defclass moves (is-a USER)
```

```
   (role concrete)
   (pattern-match reactive)
   (slot id
      (create-accessor write)
      (type INSTANCE))
   (multislot moves-list
      (create-accessor write)
      (type SYMBOL)
      (allowed-symbols no-move alone fox goat cabbage)))
 (defclass opposite-of
   (is-a USER)
   (role concrete)
   (pattern-match reactive)
   (slot value (create-accessor write))
   (slot opposite-value (create-accessor write)))
;;;*****************
;;;* INITIAL STATE *
;;;*****************
 (definstances startups
  (of status)
  (of opposite-of (value shore-1) (opposite-value shore-2))
  (of opposite-of (value shore-2) (opposite-value shore-1)))
;;;***********************
;;;* GENERATE PATH RULES *
;;;***********************
 (defrule move-alone
  ?node <- (object (is-a status)
                   (search-depth ?num)
                   (farmer-location ?fs))
  (object (is-a opposite-of) (value ?fs) (opposite-value ?ns))
  =>
  (duplicate-instance ?node
    (search-depth (+ 1 ?num))
    (parent ?node)
    (farmer-location ?ns)
    (last-move alone)))

(defrule move-with-fox
  ?node <- (object (is-a status)
                   (search-depth ?num)
                   (farmer-location ?fs)
                   (fox-location ?fs))
  (object (is-a opposite-of) (value ?fs) (opposite-value ?ns))
  =>
  (duplicate-instance ?node
    (search-depth (+ 1 ?num))
    (parent ?node)
    (farmer-location ?ns)
    (last-move fox)
    (fox-location ?ns)))
 (defrule move-with-goat
  ?node <- (object (is-a status)
                   (search-depth ?num)
                   (farmer-location ?fs)
                   (goat-location ?fs))
  (object (is-a opposite-of) (value ?fs) (opposite-value ?ns))
  =>
  (duplicate-instance ?node
    (search-depth (+ 1 ?num))
    (parent ?node)
    (farmer-location ?ns)
    (last-move goat)
    (goat-location ?ns)))
 (defrule move-with-cabbage
  ?node <- (object (is-a status)
                   (search-depth ?num)
                   (farmer-location ?fs)
                   (cabbage-location ?fs))
  (object (is-a opposite-of) (value ?fs) (opposite-value ?ns))
  =>
  (duplicate-instance ?node
    (search-depth (+ 1 ?num))
    (parent ?node)
    (farmer-location ?ns)
    (last-move cabbage)
    (cabbage-location ?ns)))
;;;******************************
```

```
;;;* CONSTRAINT VIOLATION RULES *
;;;*****************************
 (defrule fox-eats-goat
  (declare (salience 200))
  ?node <- (object (is-a status)
                   (farmer-location ?s1)
                   (fox-location ?s2&~?s1)
                   (goat-location ?s2))
  =>
  (unmake-instance ?node))
 (defrule goat-eats-cabbage
  (declare (salience 200))
  ?node <- (object (is-a status)
                   (farmer-location ?s1)
                   (goat-location ?s2&~?s1)
                   (cabbage-location ?s2))
  =>
  (unmake-instance ?node))
 (defrule circular-path
  (declare (salience 200))
  (object (is-a status)
          (search-depth ?sd1)
          (farmer-location ?fs)
          (fox-location ?xs)
          (goat-location ?gs)
          (cabbage-location ?cs))
  ?node <- (object (is-a status)
                   (search-depth ?sd2&:(< ?sd1 ?sd2))
                   (farmer-location ?fs)
                   (fox-location ?xs)
                   (goat-location ?gs)
                   (cabbage-location ?cs))
  =>
  (unmake-instance ?node))
;;;********************************
;;;* FIND AND PRINT SOLUTION RULES *
;;;********************************
 (defrule recognize-solution
  (declare (salience 100))
  ?node <- (object (is-a status)
                   (parent ?parent)
                   (farmer-location shore-2)
                   (fox-location shore-2)
                   (goat-location shore-2)
                   (cabbage-location shore-2)
                   (last-move ?move))
  =>
  (unmake-instance ?node)
  (make-instance of moves
     (id ?parent) (moves-list ?move)))
 (defrule further-solution
  (declare (salience 100))
  ?state <- (object (is-a status)
                    (parent ?parent)
                    (last-move ?move))
  ?mv <- (object (is-a moves)
                 (id ?state)
                 (moves-list $?rest))
  =>
  (modify-instance ?mv (id ?parent) (moves-list ?move ?rest)))
 (defrule print-solution
  (declare (salience 100))
  ?mv <- (object (is-a moves)
                 ;(id [no-parent])
                 (moves-list no-move $?m))
  =>
  (unmake-instance ?mv)
  (printout t t  "Solution found: " t t)
  (bind ?length (length ?m))
  (bind ?i 1)
  (bind ?shore shore-2)
  (while (<= ?i ?length)
     (bind ?thing (nth$ ?i ?m))
     (if (eq ?thing alone)
        then (printout t "Farmer moves alone to " ?shore "." t)
        else (printout t "Farmer moves with " ?thing " to " ?shore "." t))
     (if (eq ?shore shore-1)
```

```
        then (bind ?shore shore-2)
        else (bind ?shore shore-1))
     (bind ?i (+ 1 ?i))))
```

7.12.2 Sistema Experto en Vinos

```
;;;======================================================
;;;   Wine Expert Sample Problem
;;;     WINEX: The WINe EXpert system.
;;;     This example selects an appropriate wine
;;;     to drink with a meal.
;;;     CLIPS Version 6.0 Example
;;;     To execute, merely load, reset and run.
;;;======================================================
(defmodule MAIN (export ?ALL))
;;****************
;;* DEFFUNCTIONS *
;;****************
 (deffunction MAIN::ask-question (?question ?allowed-values)
   (printout t ?question)
   (bind ?answer (read))
   (if (lexemep ?answer) then (bind ?answer (lowcase ?answer)))
   (while (not (member ?answer ?allowed-values)) do
      (printout t ?question)
      (bind ?answer (read))
      (if (lexemep ?answer) then (bind ?answer (lowcase ?answer))))
   ?answer)
;;*****************
;;* INITIAL STATE *
;;*****************
(deftemplate MAIN::attribute
   (slot name)
   (slot value)
   (slot certainty (default 100.0)))
(defrule MAIN::start
  (declare (salience 10000))
  =>
  (set-fact-duplication TRUE)
  (focus QUESTIONS CHOOSE-QUALITIES WINES PRINT-RESULTS))
(defrule MAIN::combine-certainties ""
  (declare (salience 100)
           (auto-focus TRUE))
  ?rem1 <- (attribute (name ?rel) (value ?val) (certainty ?per1))
  ?rem2 <- (attribute (name ?rel) (value ?val) (certainty ?per2))
  (test (neq ?rem1 ?rem2))
  =>
  (retract ?rem1)
  (modify ?rem2 (certainty (/ (- (* 100 (+ ?per1 ?per2)) (* ?per1 ?per2)) 100))))
;;******************
;;* QUESTION RULES *
;;******************
(defmodule QUESTIONS (import MAIN ?ALL) (export ?ALL))
(deftemplate QUESTIONS::question
   (slot attribute (default ?NONE))
   (slot the-question (default ?NONE))
   (multislot valid-answers (default ?NONE))
   (slot already-asked (default FALSE))
   (multislot precursors (default ?DERIVE)))
(defrule QUESTIONS::ask-a-question
   ?f <- (question (already-asked FALSE)
                   (precursors)
                   (the-question ?the-question)
                   (attribute ?the-attribute)
                   (valid-answers $?valid-answers))
   =>
   (modify ?f (already-asked TRUE))
   (assert (attribute (name ?the-attribute)
                      (value (ask-question ?the-question ?valid-answers)))))
(defrule QUESTIONS::precursor-is-satisfied
   ?f <- (question (already-asked FALSE)
                   (precursors ?name is ?value $?rest))
         (attribute (name ?name) (value ?value))
   =>
   (if (eq (nth 1 ?rest) and)
    then (modify ?f (precursors (rest$ ?rest)))
    else (modify ?f (precursors ?rest))))
(defrule QUESTIONS::precursor-is-not-satisfied
```

```
   ?f <- (question (already-asked FALSE)
                   (precursors ?name is-not ?value $?rest))
         (attribute (name ?name) (value ~?value))
   =>
   (if (eq (nth 1 ?rest) and)
    then (modify ?f (precursors (rest$ ?rest)))
    else (modify ?f (precursors ?rest))))
;;*******************
;;* WINEX QUESTIONS *
;;*******************
(defmodule WINE-QUESTIONS (import QUESTIONS ?ALL))
(deffacts WINE-QUESTIONS::question-attributes
  (question (attribute main-component)
            (the-question "Is the main component of the meal meat, fish, or poultry? ")
            (valid-answers meat fish poultry unknown))
  (question (attribute has-turkey)
            (precursors main-component is turkey)
            (the-question "Does the meal have turkey in it? ")
            (valid-answers yes no unknown))
  (question (attribute has-sauce)
            (the-question "Does the meal have a sauce on it? ")
            (valid-answers yes no unknown))
  (question (attribute sauce)
            (precursors has-sauce is yes)
            (the-question "Is the sauce for the meal spicy, sweet, cream, or tomato? ")
            (valid-answers sauce spicy sweet cream tomato unknown))
  (question (attribute tastiness)
            (the-question "Is the flavor of the meal delicate, average, or strong? ")
            (valid-answers delicate average strong unknown))
  (question (attribute preferred-body)
            (the-question "Do you generally prefer light, medium, or full bodied wines? ")
            (valid-answers light medium full unknown))
  (question (attribute preferred-color)
            (the-question "Do you generally prefer red or white wines? ")
            (valid-answers red white unknown))
  (question (attribute preferred-sweetness)
            (the-question "Do you generally prefer dry, medium, or sweet wines? ")
            (valid-answers dry medium sweet unknown)))
;;******************
;; The RULES module
;;******************
(defmodule RULES (import MAIN ?ALL) (export ?ALL))
(deftemplate RULES::rule
  (slot certainty (default 100.0))
  (multislot if)
  (multislot then))
(defrule RULES::throw-away-ands-in-antecedent
  ?f <- (rule (if and $?rest))
  =>
  (modify ?f (if ?rest)))
(defrule RULES::throw-away-ands-in-consequent
  ?f <- (rule (then and $?rest))
  =>
  (modify ?f (then ?rest)))
(defrule RULES::remove-is-condition-when-satisfied
  ?f <- (rule (certainty ?c1)
              (if ?attribute is ?value $?rest))
  (attribute (name ?attribute)
             (value ?value)
             (certainty ?c2))
  =>
  (modify ?f (certainty (min ?c1 ?c2)) (if ?rest)))
(defrule RULES::remove-is-not-condition-when-satisfied
  ?f <- (rule (certainty ?c1)
              (if ?attribute is-not ?value $?rest))
  (attribute (name ?attribute) (value ~?value) (certainty ?c2))
  =>
  (modify ?f (certainty (min ?c1 ?c2)) (if ?rest)))
(defrule RULES::perform-rule-consequent-with-certainty
  ?f <- (rule (certainty ?c1)
              (if)
              (then ?attribute is ?value with certainty ?c2 $?rest))
  =>
  (modify ?f (then ?rest))
  (assert (attribute (name ?attribute)
                     (value ?value)
                     (certainty (/ (* ?c1 ?c2) 100)))))
```

```
(defrule RULES::perform-rule-consequent-without-certainty
  ?f <- (rule (certainty ?c1)
              (if)
              (then ?attribute is ?value $?rest))
  (test (or (eq (length$ ?rest) 0)
            (neq (nth 1 ?rest) with)))
  =>
  (modify ?f (then ?rest))
  (assert (attribute (name ?attribute) (value ?value) (certainty ?c1))))
;;*****************************
;;* CHOOSE WINE QUALITIES RULES *
;;*****************************
(defmodule CHOOSE-QUALITIES (import RULES ?ALL)
                            (import QUESTIONS ?ALL)
                            (import MAIN ?ALL))
(defrule CHOOSE-QUALITIES::startit => (focus RULES))
(deffacts the-wine-rules
;; Rules for picking the best body
  (rule (if has-sauce is yes and
            sauce is spicy)
        (then best-body is full))
  (rule (if tastiness is delicate)
        (then best-body is light))
  (rule (if tastiness is average)
        (then best-body is light with certainty 30 and
              best-body is medium with certainty 60 and
              best-body is full with certainty 30))
  (rule (if tastiness is strong)
        (then best-body is medium with certainty 40 and
              best-body is full with certainty 80))
  (rule (if has-sauce is yes and
            sauce is cream)
        (then best-body is medium with certainty 40 and
              best-body is full with certainty 60))
  (rule (if preferred-body is full)
        (then best-body is full with certainty 40))
  (rule (if preferred-body is medium)
        (then best-body is medium with certainty 40))
  (rule (if preferred-body is light)
        (then best-body is light with certainty 40))
  (rule (if preferred-body is light and
            best-body is full)
        (then best-body is medium))
  (rule (if preferred-body is full and
            best-body is light)
        (then best-body is medium))
  (rule (if preferred-body is unknown)
        (then best-body is light with certainty 20 and
              best-body is medium with certainty 20 and
              best-body is full with certainty 20))
;; Rules for picking the best color
  (rule (if main-component is meat)
        (then best-color is red with certainty 90))
 (rule (if main-component is poultry and
            has-turkey is no)
        (then best-color is white with certainty 90 and
              best-color is red with certainty 30))
 (rule (if main-component is poultry and
            has-turkey is yes)
        (then best-color is red with certainty 80 and
              best-color is white with certainty 50))
 (rule (if main-component is fish)
        (then best-color is white))
 (rule (if main-component is-not fish and
            has-sauce is yes and
            sauce is tomato)
        (then best-color is red))
 (rule (if has-sauce is yes and
            sauce is cream)
        (then best-color is white with certainty 40))
  (rule (if preferred-color is red)
        (then best-color is red with certainty 40))
  (rule (if preferred-color is white)
        (then best-color is white with certainty 40))
  (rule (if preferred-color is unknown)
        (then best-color is red with certainty 20 and
              best-color is white with certainty 20))
```

```
;; Rules for picking the best sweetness
  (rule (if has-sauce is yes and
            sauce is sweet)
        (then best-sweetness is sweet with certainty 90 and
              best-sweetness is medium with certainty 40))
  (rule (if preferred-sweetness is dry)
        (then best-sweetness is dry with certainty 40))
  (rule (if preferred-sweetness is medium)
        (then best-sweetness is medium with certainty 40))
  (rule (if preferred-sweetness is sweet)
        (then best-sweetness is sweet with certainty 40))
  (rule (if best-sweetness is sweet and
            preferred-sweetness is dry)
        (then best-sweetness is medium))
  (rule (if best-sweetness is dry and
            preferred-sweetness is sweet)
        (then best-sweetness is medium))
  (rule (if preferred-sweetness is unknown)
        (then best-sweetness is dry with certainty 20 and
              best-sweetness is medium with certainty 20 and
              best-sweetness is sweet with certainty 20))
)
;;************************
;;* WINE SELECTION RULES *
;;************************
(defmodule WINES (import MAIN ?ALL))
(deffacts any-attributes
  (attribute (name best-color) (value any))
  (attribute (name best-body) (value any))
  (attribute (name best-sweetness) (value any)))
(deftemplate WINES::wine
  (slot name (default ?NONE))
  (multislot color (default any))
  (multislot body (default any))
  (multislot sweetness (default any)))
(deffacts WINES::the-wine-list
  (wine (name Gamay) (color red) (body medium) (sweetness medium sweet))
  (wine (name Chablis) (color white) (body light) (sweetness dry))
  (wine (name Sauvignon-Blanc) (color white) (body medium) (sweetness dry))
  (wine (name Chardonnay) (color white) (body medium full) (sweetness medium dry))
  (wine (name Soave) (color white) (body light) (sweetness medium dry))
  (wine (name Riesling) (color white) (body light medium) (sweetness medium sweet))
  (wine (name Geverztraminer) (color white) (body full))
  (wine (name Chenin-Blanc) (color white) (body light) (sweetness medium sweet))
  (wine (name Valpolicella) (color red) (body light))
  (wine (name Cabernet-Sauvignon) (color red) (sweetness dry medium))
  (wine (name Zinfandel) (color red) (sweetness dry medium))
  (wine (name Pinot-Noir) (color red) (body medium) (sweetness medium))
  (wine (name Burgundy) (color red) (body full))
  (wine (name Zinfandel) (color red) (sweetness dry medium)))
(defrule WINES::generate-wines
  (wine (name ?name)
        (color $? ?c $?)
        (body $? ?b $?)
        (sweetness $? ?s $?))
  (attribute (name best-color) (value ?c) (certainty ?certainty-1))
  (attribute (name best-body) (value ?b) (certainty ?certainty-2))
  (attribute (name best-sweetness) (value ?s) (certainty ?certainty-3))
  =>
  (assert (attribute (name wine) (value ?name)
                     (certainty (min ?certainty-1 ?certainty-2 ?certainty-3)))))
;;*****************************
;;* PRINT SELECTED WINE RULES *
;;*****************************
(defmodule PRINT-RESULTS (import MAIN ?ALL))
(defrule PRINT-RESULTS::header ""
   (declare (salience 10))
   =>
   (printout t t)
   (printout t "        SELECTED WINES" t t)
   (printout t " WINE                  CERTAINTY" t)
   (printout t " -------------------------------" t)
   (assert (phase print-wines)))
(defrule PRINT-RESULTS::print-wine ""
  ?rem <- (attribute (name wine) (value ?name) (certainty ?per))
  (not (attribute (name wine) (certainty ?per1&:(> ?per1 ?per))))
  =>
```

```
  (retract ?rem)
  (format t " %-24s %2d%%%n" ?name ?per))
(defrule PRINT-RESULTS::remove-poor-wine-choices ""
  ?rem <- (attribute (name wine) (certainty ?per&:(< ?per 20)))
  =>
  (retract ?rem))
(defrule PRINT-RESULTS::end-spaces ""
   (not (attribute (name wine)))
   =>
   (printout t t))
```

7.12.3 Sistema Experto en Animales

```
;;;======================================================
;;;   Animal Identification Expert System
;;;     A simple expert system which attempts to identify
;;;     an animal based on its characteristics.
;;;     The knowledge base in this example is a
;;;     collection of facts which represent backward
;;;     chaining rules. CLIPS forward chaining rules are
;;;     then used to simulate a backward chaining inference
;;;     engine.
;;;     CLIPS Version 6.0 Example
;;;     To execute, merely load, reset, and run.
;;;     Answer questions yes or no.
;;;======================================================
;;;***************************
;;;* DEFTEMPLATE DEFINITIONS *
;;;***************************
 (deftemplate rule
   (multislot if)
   (multislot then))
;;;**************************
;;;* INFERENCE ENGINE RULES *
;;;**************************
 (defrule propagate-goal ""
   (goal is ?goal)
   (rule (if ?variable $?)
         (then ?goal ? ?value))
   =>
   (assert (goal is ?variable)))
 (defrule goal-satified ""
   (declare (salience 30))
   ?f <- (goal is ?goal)
   (variable ?goal ?value)
   (answer ? ?text ?goal)
   =>
   (retract ?f)
   (format t "%s%s%n" ?text ?value))
 (defrule remove-rule-no-match ""
   (declare (salience 20))
   (variable ?variable ?value)
   ?f <- (rule (if ?variable ? ~?value $?))
   =>
   (retract ?f))
 (defrule modify-rule-match ""
   (declare (salience 20))
   (variable ?variable ?value)
   ?f <- (rule (if ?variable ? ?value and $?rest))
   =>
   (modify ?f (if ?rest)))
 (defrule rule-satisfied ""
   (declare (salience 20))
   (variable ?variable ?value)
   ?f <- (rule (if ?variable ? ?value)
               (then ?goal ? ?goal-value))
   =>
   (retract ?f)
   (assert (variable ?goal ?goal-value)))
 (defrule ask-question-no-legalvalues ""
   (declare (salience 10))
   (not (legalanswers $?))
   ?f1 <- (goal is ?variable)
   ?f2 <- (question ?variable ? ?text)
   =>
   (retract ?f1 ?f2)
   (format t "%s " ?text)
```

```
   (assert (variable ?variable (read))))
 (defrule ask-question-legalvalues ""
   (declare (salience 10))
   (legalanswers ? $?answers)
   ?f1 <- (goal is ?variable)
   ?f2 <- (question ?variable ? ?text)
   =>
   (retract ?f1)
   (format t "%s " ?text)
   (printout t ?answers " ")
   (bind ?reply (read))
   (if (member (lowcase ?reply) ?answers)
     then (assert (variable ?variable ?reply))
          (retract ?f2)
     else (assert (goal is ?variable))))

;;;***************************
;;;* DEFFACTS KNOWLEDGE BASE *
;;;***************************
 (deffacts knowledge-base
   (goal is type.animal)
   (legalanswers are yes no)
   (rule (if backbone is yes)
         (then superphylum is backbone))
   (rule (if backbone is no)
         (then superphylum is jellyback))
   (question backbone is "Does your animal have a backbone?")
   (rule (if superphylum is backbone and
          warm.blooded is yes)
         (then phylum is warm))
   (rule (if superphylum is backbone and
          warm.blooded is no)
         (then phylum is cold))
   (question warm.blooded is "Is the animal warm blooded?")
   (rule (if superphylum is jellyback and
          live.prime.in.soil is yes)
         (then phylum is soil))
   (rule (if superphylum is jellyback and
          live.prime.in.soil is no)
         (then phylum is elsewhere))
   (question live.prime.in.soil is "Does your animal live primarily in soil?")
   (rule (if phylum is warm and
          has.breasts is yes)
         (then class is breasts))
   (rule (if phylum is warm and
          has.breasts is no)
         (then type.animal is bird/penguin))
   (question has.breasts is "Normally, does the female of your animal nurse its young with
milk?")
   (rule (if phylum is cold and
          always.in.water is yes)
         (then class is water))
   (rule (if phylum is cold and
          always.in.water is no)
         (then class is dry))
   (question always.in.water is "Is your animal always in water?")
   (rule (if phylum is soil and
          flat.bodied is yes)
         (then type.animal is flatworm))
   (rule (if phylum is soil and
          flat.bodied is no)
         (then type.animal is worm/leech))
   (question flat.bodied is "Does your animal have a flat body?")
   (rule (if phylum is elsewhere and
          body.in.segments is yes)
         (then class is segments))
   (rule (if phylum is elsewhere and
          body.in.segments is no)
         (then class is unified))
   (question body.in.segments is "Is the animals body in segments?")
   (rule (if class is breasts and
          can.eat.meat is yes)
         (then order is meat))
   (rule (if class is breasts and
          can.eat.meat is no)
         (then order is vegy))
   (question can.eat.meat is "Does your animal eat red meat?")
```

```
   (rule (if class is water and
          boney is yes)
         (then type.animal is fish))
   (rule (if class is water and
          boney is no)
         (then type.animal is shark/ray))
   (question boney is "Does your animal have a boney skeleton?")
   (rule (if class is dry and
          scally is yes)
         (then order is scales))
   (rule (if class is dry and
          scally is no)
         (then order is soft))
   (question scally is "Is your animal covered with scaled skin?")
   (rule (if class is segments and
          shell is yes)
         (then order is shell))
   (rule (if class is segments and
          shell is no)
         (then type.animal is centipede/millipede/insect))
   (question shell is "Does your animal have a shell?")
   (rule (if class is unified and
          digest.cells is yes)
         (then order is cells))
   (rule (if class is unified and
          digest.cells is no)
         (then order is stomach))
   (question digest.cells is "Does your animal use many cells to digest it's food instead
of a stomach?")
   (rule (if order is meat and
          fly is yes)
         (then type.animal is bat))
   (rule (if order is meat and
          fly is no)
         (then family is nowings))
   (question fly is "Can your animal fly?")
   (rule (if order is vegy and
          hooves is yes)
         (then family is hooves))
   (rule (if order is vegy and
          hooves is no)
         (then family is feet))
   (question hooves is "Does your animal have hooves?")
   (rule (if order is scales and
          rounded.shell is yes)
         (then type.animal is turtle))
   (rule (if order is scales and
          rounded.shell is no)
         (then family is noshell))
   (question rounded.shell is "Does the animal have a rounded shell?")
   (rule (if order is soft and
          jump is yes)
         (then type.animal is frog))
   (rule (if order is soft and
          jump is no)
         (then type.animal is salamander))
   (question jump is "Does your animal jump?")
   (rule (if order is shell and
          tail is yes)
         (then type.animal is lobster))
   (rule (if order is shell and
          tail is no)
         (then type.animal is crab))
   (question tail is "Does your animal have a tail?")
   (rule (if order is cells and
          stationary is yes)
         (then family is stationary))
   (rule (if order is cells and
          stationary is no)
         (then type.animal is jellyfish))
   (question stationary is "Is your animal attached permanently to an object?")
   (rule (if order is stomach and
          multicelled is yes)
         (then family is multicelled))
   (rule (if order is stomach and
          multicelled is no)
         (then type.animal is protozoa))
```

```
(question multicelled is "Is your animal made up of more than one cell?")
(rule (if family is nowings and
       opposing.thumb is yes)
      (then genus is thumb))
(rule (if family is nowings and
       opposing.thumb is no)
      (then genus is nothumb))
(question opposing.thumb is "Does your animal have an opposing thumb?")
(rule (if family is hooves and
       two.toes is yes)
      (then genus is twotoes))
(rule (if family is hooves and
       two.toes is no)
      (then genus is onetoe))
(question two.toes is "Does your animal stand on two toes/hooves per foot?")
(rule (if family is feet and
       live.in.water is yes)
      (then genus is water))
(rule (if family is feet and
       live.in.water is no)
      (then genus is dry))
(question live.in.water is "Does your animal live in water?")
(rule (if family is noshell and
       limbs is yes)
      (then type.animal is crocodile/alligator))
(rule (if family is noshell and
       limbs is no)
      (then type.animal is snake))
(question limbs is "Does your animal have limbs?")
(rule (if family is stationary and
       spikes is yes)
      (then type.animal is sea.anemone))
(rule (if family is stationary and
       spikes is no)
      (then type.animal is coral/sponge))
(question spikes is "Does your animal normally have spikes radiating from it's body?")
(rule (if family is multicelled and
       spiral.shell is yes)
      (then type.animal is snail))
(rule (if family is multicelled and
       spiral.shell is no)
      (then genus is noshell))
(question spiral.shell is "Does your animal have a spiral-shaped shell?")
(rule (if genus is thumb and
       prehensile.tail is yes)
      (then type.animal is monkey))
(rule (if genus is thumb and
       prehensile.tail is no)
      (then species is notail))
(question prehensile.tail is "Does your animal have a prehensile tail?")
(rule (if genus is nothumb and
       over.400 is yes)
      (then species is 400))
(rule (if genus is nothumb and
       over.400 is no)
      (then species is under400))
(question over.400 is "Does an adult normally weigh over 400 pounds?")
(rule (if genus is twotoes and
       horns is yes)
      (then species is horns))
(rule (if genus is twotoes and
       horns is no)
      (then species is nohorns))
(question horns is "Does your animal have horns?")
(rule (if genus is onetoe and
       plating is yes)
      (then type.animal is rhinoceros))
(rule (if genus is onetoe and
       plating is no)
      (then type.animal is horse/zebra))
(question plating is "Is your animal covered with a protective plating?")
(rule (if genus is water and
       hunted is yes)
      (then type.animal is whale))
(rule (if genus is water and
       hunted is no)
      (then type.animal is dolphin/porpoise))
```

```
(question hunted is "Is your animal, unfortunately, commercially hunted?")
(rule (if genus is dry and
       front.teeth is yes)
      (then species is teeth))
(rule (if genus is dry and
       front.teeth is no)
      (then species is noteeth))
(question front.teeth is "Does your animal have large front teeth?")
(rule (if genus is noshell and
       bivalve is yes)
      (then type.animal is clam/oyster))
(rule (if genus is noshell and
       bivalve is no)
      (then type.animal is squid/octopus))
(question bivalve is "Is your animal protected by two half-shells?")
(rule (if species is notail and
       nearly.hairless is yes)
      (then type.animal is man))
(rule (if species is notail and
       nearly.hairless is no)
      (then subspecies is hair))
(question nearly.hairless is "Is your animal nearly hairless?")
(rule (if species is 400 and
       land.based is yes)
      (then type.animal is bear/tiger/lion))
(rule (if species is 400 and
       land.based is no)
      (then type.animal is walrus))
(question land.based is "Is your animal land based?")
(rule (if species is under400 and
       thintail is yes)
      (then type.animal is cat))
(rule (if species is under400 and
       thintail is no)
      (then type.animal is coyote/wolf/fox/dog))
(question thintail is "Does your animal have a thin tail?")
(rule (if species is horns and
       one.horn is yes)
      (then type.animal is hippopotamus))
(rule (if species is horns and
       one.horn is no)
      (then subspecies is nohorn))
(question one.horn is "Does your animal have one horn?")
(rule (if species is nohorns and
       lives.in.desert is yes)
      (then type.animal is camel))
(rule (if species is nohorns and
       lives.in.desert is no)
      (then type.animal is giraffe))
(question lives.in.desert is "Does your animal normally live in the desert?")
(rule (if species is teeth and
       large.ears is yes)
      (then type.animal is rabbit))
(rule (if species is teeth and
       large.ears is no the type.animal is rat/mouse/squirrel/beaver/porcupine))
(question large.ears is "Does your animal have large ears?")
(rule (if species is noteeth and
       pouch is yes)
      (then type.animal is "kangaroo/koala bear"))
(rule (if species is noteeth and
       pouch is no)
      (then type.animal is mole/shrew/elephant))
(question pouch is "Does your animal have a pouch?")
(rule (if subspecies is hair and
       long.powerful.arms is yes)
      (then type.animal is orangutan/gorilla/chimpanzie))
(rule (if subspecies is hair and
       long.powerful.arms is no)
      (then type.animal is baboon))
(question long.powerful.arms is "Does your animal have long, powerful arms?")
(rule (if subspecies is nohorn and
       fleece is yes)
      (then type.animal is sheep/goat))
(rule (if subspecies is nohorn and
       fleece is no)
      (then subsubspecies is nofleece))
(question fleece is "Does your animal have fleece?")
```

```
(rule (if subsubspecies is nofleece and
       domesticated is yes)
      (then type.animal is cow))
(rule (if subsubspecies is nofleece and
       domesticated is no)
      (then type.animal is deer/moose/antelope))
(question domesticated is "Is your animal domesticated?")
(answer is "I think your animal is a " type.animal))
```

8 CAPÍTULO VIII: APRENDIZAJE DE MÁQUINA

8.1 Aprendizaje de Máquina

El aprendizaje de máquina es un campo interdisciplinario, relacionado con el desarrollo de programas de computadora que mejoran su desempeño en cierta tarea, mediante la experiencia. Los algoritmos para aprendizaje de máquina han probado su valía en una variedad de aplicaciones:

- Problemas de minería de datos.
- Dominios en los que los humanos no disponen del conocimiento necesario para desarrollar algoritmos efectivos.
- Dominios en los que los programas deben adaptarse dinámicamente a condiciones cambiantes.

Algunas disciplinas asociadas al aprendizaje de máquina, son:

- Inteligencia artificial
- Métodos Bayesianos
- Teoría de complejidad computacional
- Teoría de control
- Teoría de la información
- Filosofía
- Psicología
- Neurobiología
- Estadística

Si bien todavía no se sabe cómo hacer que los computadores aprendan tan bien como lo hacen los humanos, se ha inventado un conjunto de algoritmos que han resultado muy efectivos para ciertas tareas de aprendizaje; y, una comprensión teórica del aprendizaje está emergiendo.

A medida que la comprensión humana de los computadores vaya madurando, parece inevitable que el aprendizaje de máquina representará una importante aplicación en la ciencia y tecnología de computación.

Varios programas de computadora desarrollados muestran útiles formas de aprendizaje y significativas aplicaciones comerciales han aparecido:

- Reconocimiento del habla en diversos idiomas,
- Minería de datos empresariales, aplicaciones de crédito, transacciones financieras, fichas médicas, etc.

- Conducción de un vehículo autónomo
- Clasificación de nuevas estructuras astronómicas
- Juegos (Ajedrez, Backgammon).

El diseño de aplicaciones de aprendizaje de máquina implica un conjunto de decisiones:

- El tipo de experiencia de entrenamiento
- La función objetivo a ser aprendida
- Una representación para la función objetivo
- Un algoritmo para al aprendizaje de la función objetivo, a partir de ejemplos de entrenamiento

En esencia, el aprendizaje involucra un proceso de búsqueda en el espacio de hipótesis, para encontrar la que mejor se ajuste a los ejemplos de entrenamiento y otras restricciones o conocimiento previo.

8.2 Aprendizaje de Conceptos

Un programa de computador se dice que aprende de una experiencia ***E*** con respecto a algún tipo de clase de tareas ***T***, con cierta medida de desempeño ***D***, si su respuesta a las tareas en ***T*** mejora dada la experiencia ***E***, de acuerdo con la medición establecida por ***D***.

Para que un problema de aprendizaje esté bien definido, es necesario identificar tres atributos:

- La clase de tareas ***T*** a ser aprendidas;
- La medida de desempeño ***D*** a ser mejorada; y,
- La fuente de donde derivar las experiencias ***E***.

8.2.1 Problema de aprender a jugar damas

- Tarea ***T***: Jugar damas
- Medida de desempeño ***D***: Porcentaje de juegos ganados a oponentes
- Experiencia de entrenamiento ***E***: Practicar juegos consigo mismo o con oponentes

8.2.2 Problema de aprender a reconocer palabras manuscritas

- Tarea ***T***: Reconocer y clasificar palabras manuscritas dentro de imágenes
- Medida de desempeño ***D***: Porcentaje de palabras clasificadas correctamente
- Experiencia de entrenamiento ***E***: Base de datos de palabras manuscritas debidamente clasificadas

8.2.3 Problema de enseñar a un robot a conducir un vehículo

- Tarea ***T***: Conducir un vehículo en una carretera pública de 4 carriles utilizando visión robótica

- Medida de desempeño ***D***: Promedio de distancia viajada antes de que un supervisor humano detecte un error
- Experiencia de entrenamiento ***E***: Secuencia de imágenes y comandos de control grabados durante la operación de un conductor humano

8.3 Diseño de un Sistema de Aprendizaje

La Experiencia de Entrenamiento.- La primera acción es escoger el tipo de experiencia de entrenamiento que va a hacer que el sistema aprenda. El tipo de experiencia de entrenamiento disponible puede tener un impacto significativo en el éxito o fracaso del aprendiz.

El segundo atributo importante de la experiencia de entrenamiento es el grado al cual el aprendiz controla la secuencia de los ejemplos de entrenamiento.

El tercer atributo importante de la experiencia de entrenamiento es que tan bien representa la distribución de ejemplos sobre la cual el desempeño final D del sistema debe ser medido.

La Función Objetivo.- El siguiente paso es determinar exactamente qué tipo de conocimiento será aprendido y cómo este será utilizado por el programa de desempeño.

Para esto se requiere definir una función objetivo capaz de aprender. La función objetivo debe asignar un valor numérico a cada una de las opciones de solución que puedan ser aplicables en un determinado estado del proceso de aprendizaje. A los mejores estados la función objetivo asignará los valores más altos.

Representación de la Función Objetivo.- Una vez que se ha especificado la función objetivo ideal, se debe escoger una representación que el programa de aprendizaje usará para describir la función que deberá aproximar durante el entrenamiento.

El tipo de función puede ser polinomial, lineal o no lineal, como la obtenida a partir de una red neuronal artificial.

Algoritmo de Aproximación a la Función Objetivo.- A fin de configurar la función objetivo, se requiere un conjunto de ejemplos para el entrenamiento.

Cada ejemplo de entrenamiento es un par ordenado de conformado por una instancia de la función objetivo y la respuesta esperada para la misma.

A partir de los ejemplos, durante el entrenamiento, se van estimando los valores con la ayuda de un algoritmo de aprendizaje.

Diseño Final.- El diseño final del sistema de aprendizaje puede ser descrito mediante cuatro programas modulares, que representan los componentes principales de los sistemas de aprendizaje:



Figura 45. Módulos de un Sistema de Aprendizaje

- **El Sistema de Desempeño**.- Este módulo debe resolver la tarea establecida para el sistema de aprendizaje. Su operación está guiada por la función objetivo aprendida.
- **El Crítico**.- Toma como entrada la historia de acciones del sistema y produce como salida un conjunto de ejemplos de entrenamiento de la función objetivo.
- **El Generalizador**.- Toma como entrada los ejemplos de entrenamiento y produce una hipótesis de salida que es una estimación de la función objetivo.
- **El Generador Experimental**.- Toma como entrada la hipótesis actual y entrega una nueva opción para que el sistema de desempeño la explore.

8.4 Minería de Datos

La minería de datos se la puede definir como el proceso de descubrir patrones útiles en grandes cantidades de datos, ya sea automática o semiautomáticamente. [130]

Para que un patrón sea de utilidad, debe permitir la realización de predicciones y la solución de problemas mediante la extracción de información obtenida de las bases de datos.

Los patrones deben ser representados en términos de una estructura que pueda ser examinada, permita el razonamiento y genere información para apoyar decisiones. A estos **patrones** se los denomina **estructurales** ya que capturan la estructura de la decisión y permiten explicar algo acerca de los datos. Muchas de estas técnicas han sido desarrolladas dentro del campo del aprendizaje de máquina.

La minería de datos es un tópico que involucra el aprendizaje desde una perspectiva práctica, antes que teórica. El interés se centra en la incorporación de técnicas para encontrar y describir **patrones estructurales** en los datos, y que sirvan como herramientas para entender los datos, ganar conocimiento y realizar predicciones. En los procesos de minería, los datos tienen la forma de conjuntos de ejemplos y las salidas tienen la forma de predicciones acerca de nuevos ejemplos. Las salidas también pueden incluir una

---

[130] Witten I H, Frank E, Hall M A. *DATA MINING, Practical Machine Learning Tools and Techniques*, 3rd Ed. Morgan Kaufmann Publishers, Burlington, MA 01803, USA, 2011

descripción de la estructura que podría ser utilizada tanto para identificar ejemplos cuya clase se desconoce, como para ganar conocimiento a partir de los datos.

Muchas de las técnicas de aprendizaje buscan descripciones estructurales de lo que se ha aprendido. Estas descripciones, típicamente pueden ser expresadas como un conjunto de reglas o como árboles de decisión.

8.5 Reglas

Existen diferentes formas de identificar reglas de clasificación, a partir de una muestra de ejemplos.

8.5.1 Método 1-R

Es uno de los métodos más simples, que genera un conjunto de reglas que prueban un atributo en particular. 1-R escoge el atributo que produce reglas con el menor número de errores. El pseudocódigo es el siguiente:

```
Para cada atributo:
      Para cada valor del atributo:
            Contar las veces que cada clase aparece
            Identificar la clase más frecuente
            Hacer que la regla asigne esa clase para el valor del
            atributo
      Calcular la taza de error de la regla
Escoger la regla con la menor tasa de error.
```

Para ilustrar el método 1-R, se toman los datos del ejemplo ***weather.nominal.arff***, dado en la herramienta **WEKA**. [131]

Pronóstico	Temperatura	Humedad	Ventoso	¿Juega?
soleado	caliente	alta	FALSO	no
soleado	caliente	alta	VERDADERO	no
nublado	caliente	alta	FALSO	si
lluvioso	templada	alta	FALSO	si
lluvioso	fresca	normal	FALSO	si
lluvioso	fresca	normal	VERDADERO	no
nublado	fresca	normal	VERDADERO	si
soleado	templada	alta	FALSO	no
soleado	fresca	normal	FALSO	si
lluvioso	templada	normal	FALSO	si
soleado	templada	normal	VERDADERO	si
nublado	templada	alta	VERDADERO	si
nublado	caliente	normal	FALSO	si
lluvioso	templada	alta	VERDADERO	no

[131] http://www.cs.waikato.ac.nz/ml/weka/downloading.html

La tabla indica las condiciones ambientales que tienen que satisfacerse para que una persona decida jugar tenis. De aquí, aplicando el pseudocódigo del método 1-R, se obtienen las siguientes reglas con sus respectivas tazas de error.

N°	Atributo	Valor	¿Juega?	Errores	Total Errores
1	pronóstico	soleado	no	2/5	4/14
		nublado	si	0/4	
		lluvioso	si	2/5	
2	temperatura	caliente	no	2/4	5/14
		templada	si	2/6	
		fresca	si	1/4	
3	humedad	alta	no	3/7	4/14
		normal	si	1/7	
4	ventoso	FALSO	si	2/8	5/14
		VERDADERO	no	3/6	

Como las reglas 1 y 3 tienen las menores tazas de error, pero son iguales, se escoge aleatoriamente una de ellas. Si se aplica cualquiera de las reglas, ya sea la 1 o la 3 a los datos dados, se puede comprobar que se contabilizan 4 errores en los 14 ejemplos.

8.5.2 Método PRISM

El método PRISM es un **algoritmo de cobertura** que va añadiendo pruebas a la regla que se está construyendo, siempre con el objetivo de crear una regla con la máxima precisión. Para ello, se escoge un par atributo – valor, para maximizar la probabilidad de la clasificación deseada.

El pseudocódigo del método PRISM para construir reglas, se indica a continuación:

```
Para cada clase C
  Inicializar E a la instancia establecida
  Mientras que E contiene instancias de la clase C
    Crear una regla R con un lado izquierdo (LHS) vacío que prediga
     la clase C
    Hasta que R sea perfecto (o no haya más atributos que se pueda
     utilizar) hacer
      Para cada atributo A no mencionado en R, y cada valor v,
        Considere añadir la condición A = v al LHS de R
        Seleccionar A y v para maximizar la precisión p / t
        (En caso de empate, elegir la condición con la mayor p)
      Añadir A = v en R
    Retirar los casos cubiertos por R de E
```

Para ilustrar su operación, se toma como ejemplo los datos del archivo ***contact-lenses.arff***, incluido entre los casos de la herramienta **WEKA**.

N°	Edad	Diagnóstico	Astigmatismo	Producción de Lágrimas	Recomendación Lentes de Contacto
1	joven	miopía	no	reducida	ninguno
2	joven	miopía	no	normal	suave
3	joven	miopía	si	reducida	ninguno
4	joven	miopía	si	normal	duro
5	joven	hipermetropía	no	reducida	ninguno
6	joven	hipermetropía	no	normal	suave
7	joven	hipermetropía	si	reducida	ninguno
8	joven	hipermetropía	si	normal	duro
9	pre-présbita	miopía	no	reducida	ninguno
10	pre-présbita	miopía	no	normal	suave
11	pre-présbita	miopía	si	reducida	ninguno
12	pre-présbita	miopía	si	normal	duro
13	pre-présbita	hipermetropía	no	reducida	ninguno
14	pre-présbita	hipermetropía	no	normal	suave
15	pre-présbita	hipermetropía	si	reducida	ninguno
16	pre-présbita	hipermetropía	si	normal	ninguno
17	présbita	miopía	no	reducida	ninguno
18	présbita	miopía	no	normal	ninguno
19	présbita	miopía	si	reducida	ninguno
20	présbita	miopía	si	normal	duro
21	présbita	hipermetropía	no	reducida	ninguno
22	présbita	hipermetropía	no	normal	suave
23	présbita	hipermetropía	si	reducida	ninguno
24	présbita	hipermetropía	si	normal	ninguno

Se inicia la construcción de la regla para la recomendación de lentes de contacto duros.

Si ? entonces recomendación = duro

Para el término **?** desconocido, se tienen 9 opciones:

Edad = joven	2/8
Edad = pre-présbita	1/8
Edad = présbita	1/8
Diagnóstico = miopía	3/12
Diagnóstico = hipermetropía	1/12
Astigmatismo = no	0/12
Astigmatismo = si	4/12
Producción de Lágrimas = reducida	0/12
Producción de Lágrimas = normal	4/12

De aquí se escoge la tasa más alta, en este caso como hay 2, se toma la primera. De esta manera, la regla queda:

Si astigmatismo = si entonces la recomendación = duro

Como la precisión es apenas 4/12, se refina la regla:

Si astigmatismo = si y ? entonces la recomendación = duro

Ahora, el término ? desconocido tiene las siguientes siete opciones:

Edad = joven	2/4
Edad = pre-présbita	1/4
Edad = présbita	1/4
Diagnóstico = miopía	3/6
Diagnóstico = hipermetropía	1/6
Producción de Lágrimas = reducida	0/6
Producción de Lágrimas = normal	4/6

Con estos resultados, la regla queda:

Si astigmatismo = si y producción de lágrimas = normal
Entonces la recomendación = duro

Un nuevo refinamiento a la regla, produce:

Si astigmatismo = si y producción de lágrimas = normal y ?
Entonces la recomendación = duro

Las opciones para el término desconocido, son:

Edad = joven	2/2
Edad = pre-présbita	1/2
Edad = présbita	1/2
Diagnóstico = miopía	3/3
Diagnóstico = hipermetropía	1/3

Al seleccionar la opción 4 por tener mayor cobertura (3/3), genera la primera regla:

Si astigmatismo = si y producción de lágrimas = normal y diagnóstico = miopía
Entonces la recomendación = duro

Al aplicar esta regla al conjunto de ejemplos, se puede ver que cubre tres de los cuatro casos en los que se recomienda lentes de contacto duros, por lo que se remueven estos casos del conjunto (4, 12 y 20) y se inicia de nuevo el proceso para encontrar otra regla del tipo:

Si ? entonces recomendación = duro

La única posibilidad mayor a cero, en este caso es:

Edad = joven	1/7

A continuación se busca otro término:

Si edad = joven y ? entonces recomendación = duro

Para el segundo término, la mejor opción es:

Astigmatismo = si	**1/3**

Y para el tercer término, resulta:

Producción de Lágrimas = normal	**1/1**

Con lo que la segunda regla queda así:

Si edad = joven y astigmatismo = si y producción de lágrimas = normal
Entonces la recomendación = duro

Ahora que todos los casos de recomendación de lentes de contacto duros están cubiertos, se procede de igual manera para encontrar las reglas para el caso de lentes de contacto suaves y el de recomendación igual a ninguno. El conjunto de reglas resultante, se muestra a continuación:

Si astigmatismo = si
y producción de lágrimas = normal
y diagnóstico = miopía entonces recomendación = duro
Si edad = joven
y astigmatismo = si
y producción de lágrimas = normal entonces recomendación = duro
Si astigmatismo = no
y producción de lágrimas = normal
y diagnóstico = hipermetropía entonces recomendación = suave
Si astigmatismo = no
y producción de lágrimas = normal
y edad = joven entonces recomendación = suave
Si edad = pre-présbita
y astigmatismo = no
y producción de lágrimas = normal entonces recomendación = suave
Si producción de lágrimas = reducida entonces recomendación = ninguno
Si edad = présbita
y producción de lágrimas = normal
y diagnóstico = miopía
y astigmatismo = no entonces recomendación = ninguno
Si diagnóstico = hipermetropía
y astigmatismo = si
y edad = pre-présbita entonces recomendación = ninguno
Si edad = présbita
y diagnóstico = hipermetropía
y astigmatismo = si entonces recomendación = ninguno

8.6 Modelación Estadística

El método 1-R utiliza un solo atributo como base para las decisiones y escoge el que mejor funciona. Otra técnica simple es utilizar todos los atributos y permitir que contribuyan a la decisión, asumiendo que son igualmente importantes e independientes entre sí.

Tomando los datos del ejemplo ***weather.nominal.arff***, se pueden obtener las estadísticas que se resumen en la siguiente Tabla:

Pronóstico			Temperatura			Humedad			Ventoso			¿Juega?	
¿Juega?	SI	NO	¿Juega?	SI	NO	¿Juega?	SI	NO	¿Juega?	SI	NO	SI	NO
Soleado	2	3	**Caliente**	2	2	**Alta**	3	4	**FALSO**	6	2	9	5
Nublado	4	0	**Templada**	4	2	**Normal**	6	1	**VERDADERO**	3	3		
Lluvioso	3	2	**Fresca**	3	1								
Soleado	**0,222**	**0,6**	**Caliente**	**0,222**	**0,4**	**Alta**	**0,333**	**0,8**	**FALSO**	**0,667**	**0,4**	**0,643**	**0,357**
Nublado	**0,444**	**0**	**Templada**	**0,444**	**0,4**	**Normal**	**0,667**	**0,2**	**VERDADERO**	**0,333**	**0,6**		
Lluvioso	**0,333**	**0,4**	**Fresca**	**0,333**	**0,2**								

Dado un caso como el siguiente, se puede calcular la posibilidad para salir a jugar = Si y salir a Jugar = No:

Pronóstico	Temperatura	Humedad	Ventoso	¿Juega?
Soleado	Fresca	Alta	VERDADERO	?

Posibilidad de jugar = SI:

0,2222	0,3333	0,3333	0,3333	0,6429	**0,0053**

Posibilidad de jugar = NO:

0,6000	0,2000	0,8000	0,6000	0,3571	**0,0206**

Normalizando las posibilidades, se convierten en probabilidades:

$$Probabilidad\ de\ Jugar = SI: \frac{0{,}0053}{0{,}0053 + 0{,}0206} = 20{,}5\%$$

$$Probabilidad\ de\ Jugar = NO: \frac{0{,}0206}{0{,}0053 + 0{,}0206} = 79{,}5\%$$

Los resultados indican que lo más probable es que NO salga a jugar. Este método simple e intuitivo está basado en la **Regla de probabilidad condicional de Bayes**. Debido a las consideraciones subyacentes, al método se lo conoce como Naive Bayes.

8.7 Árboles de Decisión

El aprendizaje por medio de árboles de decisión, es uno de los métodos prácticos más ampliamente utilizados para la inferencia por inducción. [132]

Los árboles de decisión clasifican instancias, ordenándolas a partir de la raíz, recorriendo diversos nodos de decisión, hasta llegar a un nodo tipo hoja, que es el que proporciona la clasificación de la instancia considerada. Cada nodo de decisión especifica la prueba de algún atributo de la instancia y cada ramal descendiente que parte del nodo corresponde a uno de los valores posibles que puede asumir el atributo.

[132] Mitchell, T. M. *Machine Learning*. MIT Press and McGraw-Hill, USA, 1997.

El aprendizaje por medio de árboles de decisión es generalmente más apropiado para problemas que poseen las siguientes características:

- Las instancias son representadas por pares atributo – valor.
- La función objetivo posee valores de salida discretos.
- Las descripciones son disyuntivas.
- Los datos de entrenamiento, pueden contener errores.
- Los datos de entrenamiento pueden tener atributos con valores incompletos o faltantes.

La mayoría de los algoritmos desarrollados para el aprendizaje de árboles de decisión, son variantes alrededor del algoritmo central que utiliza la búsqueda codiciosa de arriba hacia abajo (*Top-down greedy search*) a través del espacio de posibles árboles de decisión. Esta estrategia de búsqueda, está ejemplificada en el algoritmo ID3[133] y su sucesor, el C4.5[134].

8.7.1 Algoritmo ID3

El algoritmo ID3 básico, empieza la construcción del árbol de decisión, con la pregunta:

¿**Cuál atributo debería ser comprobado en la raíz del árbol**?

Para contestar esta pregunta, cada atributo de las instancias es evaluado utilizando una prueba estadística para determinar qué tan bien, por sí solo, es capaz de clasificar los ejemplos de entrenamiento. Se selecciona el mejor atributo y se lo usa como la prueba en la raíz del árbol. Luego se crea un descendiente de la raíz del árbol por cada posible valor de este atributo y los ejemplos de entrenamiento se ordenan en el nodo descendiente apropiado. Luego se repite todo el proceso utilizando los ejemplos de entrenamiento asociados con cada nodo descendiente a fin de seleccionar el mejor atributo a probarse en este nivel del árbol.

La medida cuantitativa que indica la propiedad clasificadora de un atributo, es la **ganancia de información**. La ganancia de información se define como la reducción esperada en la entropía, causada por la partición de los ejemplos, de acuerdo con el atributo considerado.

La entropía es una medición de la homogeneidad de los ejemplos. Dada una colección ***S***, conteniendo ejemplos de algún concepto objetivo cuyo atributo puede tomar ***c*** valores diferentes, la entropía de ***S*** en relación a la clasificación de orden ***c***, está representada por:

$$Entropía(S) \equiv \sum_{i=1}^{c} -p_i \, log_2 \, p_i$$

---

[133] Quinlan, J. R. *Induction of Decision Trees*. Machine Learning 1: 81-186. Kluwer Academic Publishers, Boston, 1986.
[134] Quinlan, J. R. *C4.5: Programs for Machine Learning*. San Mateo, CA; Morgan Kaufmann, 1993.

Dada esta relación, se puede definir la ganancia de información de un atributo ***A***, relativo a la colección de ejemplos ***S***, como:

$$Ganancia(S, A) \equiv Entropía(S) - \sum_{v \in Valores(A)} \frac{|S_v|}{|S|} Entropía(S_v)$$

Para ilustrar la operación del algoritmo ID3 básico, se toma nuevamente la colección de ejemplos ***weather.nominal.arff***, dado en la herramienta **WEKA**:

	Pronóstico	Temperatura	Humedad	Ventoso	¿Juega?
D1	soleado	caliente	alta	FALSO	no
D2	soleado	caliente	alta	VERDADERO	no
D3	nublado	caliente	alta	FALSO	si
D4	lluvioso	templada	alta	FALSO	si
D5	lluvioso	fresca	normal	FALSO	si
D6	lluvioso	fresca	normal	VERDADERO	no
D7	nublado	fresca	normal	VERDADERO	si
D8	soleado	templada	alta	FALSO	no
D9	soleado	fresca	normal	FALSO	si
D10	lluvioso	templada	normal	FALSO	si
D11	soleado	templada	normal	VERDADERO	si
D12	nublado	templada	alta	VERDADERO	si
D13	nublado	caliente	normal	FALSO	si
D14	lluvioso	templada	alta	VERDADERO	no

Se inicia el análisis de los ejemplos, para encontrar el mejor atributo a ser comprobado en la raíz del árbol de decisión. Las operaciones de entropía y ganancia de entropía para cada atributo, están tabuladas a continuación. De los resultados obtenidos se puede determinar que el atributo **Pronóstico** proporciona la mejor predicción en el nodo raíz, con una ganancia de entropía de **0,2467**.

		S_v/S*Entropía(S_v)			
Atributo	**Entropía(S)**	**Soleado**	**Nublado**	**Lluvioso**	**Ganancia**
Pronóstico	0,9403	0,3468	0,0000	0,3468	**0,2467**

Atributo	**Entropía(S)**	**Caliente**	**Templada**	**Fresca**	**Ganancia**
Temperatura	0,9403	0,2857	0,3936	0,2318	**0,0292**

		S_v/S*Entropía(S_v)		
Atributo	**Entropía(S)**	**Alta**	**Normal**	**Ganancia**
Humedad	0,9403	0,4926	0,2958	**0,1518**

Atributo	**Entropía(S)**	**VERDADERO**	**FALSO**	**Ganancia**
Ventoso	0,9403	0,4286	0,4636	**0,0481**

El árbol parcial resultante, se muestra en la siguiente figura:

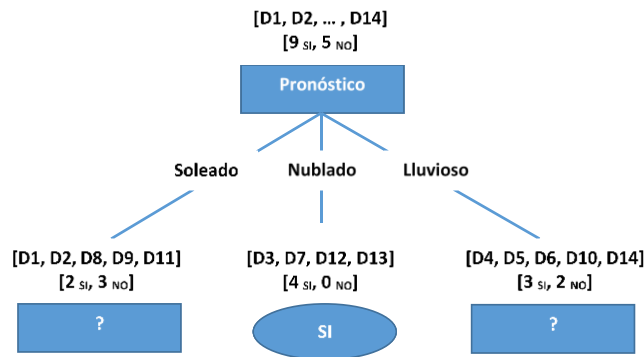


Figura 46. Árbol de decisión parcial para ejemplo *weather.nominal.arff*

Nótese que el ramal de valor igual **Nublado** del atributo **Pronóstico** ha alcanzado un valor de entropía igual a cero, por lo que se llega a un nodo tipo hoja, con la clasificación igual a **juega = SI**.

El proceso de seleccionar un nuevo atributo se repite para cada nodo descendiente, utilizando únicamente los ejemplos de entrenamiento asociados con dichos nodos.

Para el caso de **Pronóstico = soleado**, la tabla de resultados indica que el mejor atributo es la humedad, que con sus valores **humedad = alta**, resuelve a **juega = NO**; y, con **humedad = normal**, resuelve a **juega = SI**.

		S_v/S*Entropía(S_v)			
Atributo	**Entropía(S_{Sol})**	**Caliente**	**Templada**	**Fresca**	**Ganancia**
Temperatura	0,9710	0,0000	0,4000	0,0000	**0,5710**

		S_v/S*Entropía(S_v)		
Atributo	**Entropía(S_{Sol})**	**Alta**	**Normal**	**Ganancia**
Humedad	0,9710	0,0000	0,0000	**0,9710**

Atributo	**Entropía(S_{Sol})**	**VERDADERO**	**FALSO**	**Ganancia**
Ventoso	0,9710	0,4000	0,5510	**0,0200**

Para el caso de **Pronóstico = lluvioso**, la siguiente tabla de resultados indica que el mejor atributo es el que indica el estado del viento. Si **ventoso = FALSO**, resuelve a **juega = SI**; y, con **ventoso = VERDADERO**, resuelve a **juega = NO**.

		S_v/S*Entropía(S_v)			
Atributo	**Entropía(S_{Lluv})**	**Caliente**	**Templada**	**Fresca**	**Ganancia**
Temperatura	0,9710	0,0000	0,5510	0,4000	**0,0200**

		S_v/S*Entropía(S_v)		
Atributo	**Entropía(S_{Lluvi})**	**VERDADERO**	**FALSO**	**Ganancia**
Ventoso	0,9710	0,0000	0,0000	**0,9710**

Con estos resultados, el árbol de decisión para los datos dados en el ejemplo, queda como se indica en la siguiente figura:

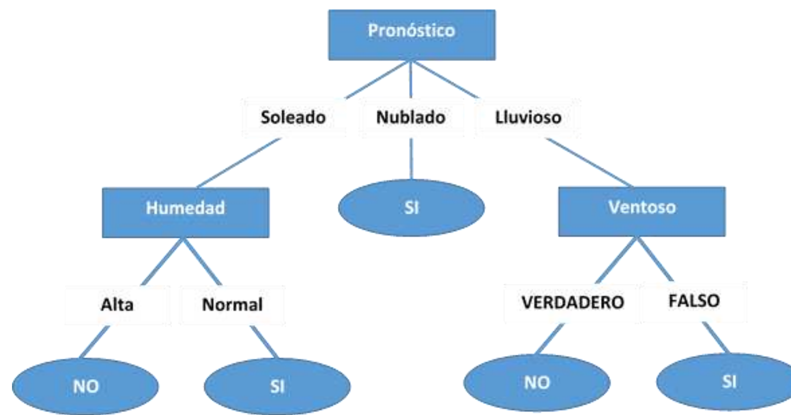


Figura 47. Árbol de Decisión completo para datos de ejemplo dado en: *weather.nominal.arff*

Algunos problemas prácticos que aparecen en el aprendizaje basado en árboles de decisión, incluyen:

- Qué tan profundo se debe construir el árbol de decisión.
- Cómo se debe manejar atributos cuyos valores no sean nominales sino continuos.
- Cómo seleccionar una métrica apropiada para la selección de atributos.
- Manejar ejemplos de entrenamiento cuyos atributos tienen datos faltantes o están incompletos.
- Administrar atributos con diferentes costos o ponderaciones.
- Mejorar la eficiencia computacional de los algoritmos.

8.7.2 Algoritmo C4.5

Una amplia variedad de extensiones al algoritmo ID3 básico han sido desarrolladas por diferentes investigadores. Entre estas se incluyen métodos para la poda de árboles, el manejo de atributos con valores reales, procesamiento de ejemplos de entrenamiento que tienen valores faltantes en sus atributos, refinamiento incremental de árboles de decisión a medida que se dispone de nuevos ejemplos de entrenamiento, uso de otras métricas para la selección de atributos diferentes de la ganancia de información, consideración de costos asociados a las instancias de los atributos. Uno de los algoritmos que considera la mayoría de estas extensiones, es el denominado C4.5. [135]

8.8 Herramientas para Aprendizaje de Máquina

En los últimos años la comunidad de investigadores se ha beneficiado con la disponibilidad de numerosas herramientas tipo ***open source***. Ofrecen implementaciones eficientes de algoritmos de amplia aplicación. En esta sección se presentan algunas de estas herramientas disponibles.

[135] Quinlan, J. R. *C4.5: Programs for Machine Learning*. San Mateo, CA; Morgan Kaufmann, 1993.

8.8.1 DARWIN

El marco de trabajo **DARWIN** ha sido desarrollado para cumplir con dos metas: en primer lugar, proporcionar una biblioteca estable, robusta y eficiente de aprendizaje automático para los profesionales; y, en segundo lugar, proporcionar la infraestructura para que estudiantes e investigadores puedan experimentar y extender los métodos del estado del arte. Con este fin, proporciona la infraestructura de gestión de datos, registro y configuración, una interfaz coherente con el aprendizaje de máquina estándar, modelos probabilísticos y algoritmos gráficos. El código fuente está bien documentado y es fácil de extender. El marco de trabajo DARWIN se distribuye bajo la licencia BSD. Es importante destacar que es gratuito tanto para uso académico como para comercial.[136]

8.8.2 Dlib-ml

Dlib-ml es una biblioteca de código abierto, dirigida tanto para ingenieros como para investigadores científicos. Su objetivo es proporcionar un ambiente rico para el desarrollo de software de aprendizaje de máquina en el lenguaje C++. Con este fin, dlib- ml contiene un conjunto de herramientas de álgebra lineal extensible con soporte incorporado BLAS. También cuenta con las implementaciones de algoritmos para realizar inferencia en redes bayesianas y métodos basados en *kernel* de clasificación, regresión, *clustering*, detección de anomalías, y *ranking*. Para habilitar el uso fácil de estas herramientas, la biblioteca entera ha sido desarrollada con una documentación completa y precisa. Cuenta también con poderosas herramientas de depuración. [137]

8.8.3 GPML-Toolbox

La caja de herramientas **GPML** ofrece una amplia gama de funcionalidades para procesos Gaussianos (GP) de inferencia y predicción. Los GPs son especificados por las funciones media y covarianza. Se ofrece una biblioteca de funciones y mecanismos de media y covarianza simples para componer funciones más complejas. Varias funciones de verosimilitud son soportadas, como Gaussianas y de cola pesada para la regresión, así como otras apropiadas para clasificación. Por último, se proporciona una gama de métodos de inferencia, incluyendo inferencia exacta y variacional, expectativa de propagación, y el método de Laplace para tratar con probabilidades no gaussianas y FITC para hacer frente a las tareas de regresión grandes. [138]

8.8.4 Java-ML

Java-ML es una colección de algoritmos de aprendizaje automático y minería de datos, que tiene como objetivo ser un API de fácil utilización y extensible para desarrolladores de software e investigadores científicos.

---

[136] http://drwn.anu.edu.au/
[137] http://dlib.net/ml.html
[138] http://gaussianprocess.org/gpml/code/matlab/doc/index.html

Las interfaces para cada tipo de algoritmo se mantienen simples y los algoritmos siguen estrictamente su interfaz respectiva. Por lo tanto, la comparación de diferentes clasificadores o algoritmos de *clustering* es sencillo, y la implementación de nuevos algoritmos también es fácil. Las implementaciones de los algoritmos están claramente escritas, debidamente documentadas y por lo tanto se pueden utilizar como una referencia. La biblioteca está escrita en Java y está disponible bajo la licencia GNU GPL. [139]

8.8.5 MLPACK

MLPACK es una librería estado-del-arte de aprendizaje de máquina, escalable, multi-plataforma escrita en C++. Se lanzó a finales de 2011 y ofrece tanto una API simple, consistente, accesible a usuarios noveles, pero de alto rendimiento y flexibilidad para los usuarios expertos, aprovechando las características modernas del lenguaje C++. MLPACK proporciona algoritmos de vanguardia cuyos puntos de referencia muestran un rendimiento mucho mejor que otras bibliotecas de aprendizaje de máquina. MLPACK versión 1.0.3, es licenciado bajo la LGPL, en: **http://www.mlpack.org**

8.8.6 Scikit-learn

Scikit-learn es un módulo en Python que integra una amplia gama de algoritmos de aprendizaje de máquina del estado del arte, para problemas medianos supervisados y no supervisados. Este paquete se centra en ofrecer aprendizaje de máquina a los no especialistas incorporando el uso de un lenguaje de alto nivel de propósito general. Se hace énfasis en la facilidad de uso, rendimiento, documentación, y la coherencia de la API. Cuenta con dependencias mínimas y se distribuye bajo la licencia BSD simplificada, fomentando su uso tanto en entornos académicos y comerciales. El código fuente, los binarios y la documentación se puede descargar de: **http://scikit-learn.sourceforge.net**

8.8.7 SHOGUN

SHOGUN, que está diseñado para el aprendizaje unificado a gran escala para una amplia gama de tipos de fenómenos y situaciones de aprendizaje. Ofrece un gran número de modelos de aprendizaje de máquina tales como máquinas de soporte vectorial, modelos ocultos de Markov, aprendizaje múltiple núcleo, análisis discriminante lineal, y más. La mayoría de los algoritmos específicos son capaces de tratar con varias clases de datos diferentes. Esta caja de herramientas se ha utilizado en varias aplicaciones de biología computacional, algunas de ellas vienen con no menos de 50 millones de ejemplos de entrenamiento y otras con 7 mil millones de ejemplos de prueba. Con más de mil instalaciones en todo el mundo, SHOGUN ya ha sido ampliamente adoptado en la comunidad de aprendizaje automático y más allá. SHOGUN está implementado en C + + y tiene interfaces con MATLAB, R, Octave, Python, y tiene una interfaz de línea de comandos

[139] **http://java-ml.sourceforge.net/**

independiente. El código fuente está disponible libremente bajo la Licencia Pública General de GNU, en **http://www.shogun-toolbox.org**

8.8.8 Waffles

Waffles es un conjunto de herramientas que buscan brindar una amplia diversidad de operaciones útiles en el aprendizaje de máquina y campos relacionados, sin imponer procesos innecesarios o restricciones de una interfaz de usuario. Esto se hace proporcionando una sencilla interfaz de línea de comandos (CLI), que realizan tareas básicas. La CLI es ideal para este propósito, ya que está bien establecida, está disponible en los sistemas operativos más comunes, y es accesible a través de la mayoría de los lenguajes de programación. Dado que estas herramientas realizan operaciones en un nivel bastante granular, se pueden utilizar en formas no previstas por el diseñador de la interfaz. La última versión se puede descargar desde **http://waffles.sourceforge.net**. También se puede encontrar en ese sitio la documentación completa de las herramientas de la CLI, incluyendo muchos ejemplos, así como documentación para los desarrolladores que buscan vincular con la biblioteca GClasses.

8.8.9 WEKA

Weka es una colección de algoritmos de aprendizaje automático para tareas de minería de datos. Los algoritmos bien se pueden aplicar directamente a un conjunto de datos o ser llamados desde su propio código Java. Weka contiene herramientas para pre-procesamiento de los datos, clasificación, regresión, clustering, reglas de asociación, y la visualización. También es muy adecuado para el desarrollo de nuevos sistemas de aprendizaje de máquina.

Weka es un software de código abierto publicado bajo la Licencia Pública General GNU. Está disponible en: **http://www.cs.waikato.ac.nz/ml/weka/**

9 CAPÍTULO IX: COMPUTACIÓN NEURONAL

9.1 Introducción

La visión actual del sistema nervioso debe mucho a las contribuciones de dos eminentes científicos: **Santiago Ramón y Cajal**[140], un médico español, especializado en histología y anátomo-patología microscópica, que obtuvo el premio Nobel de Medicina en 1906 por descubrir los mecanismos que gobiernan la morfología y los procesos conectivos de las células nerviosas, una revolucionaria teoría basada en que el tejido cerebral está compuesto por células individuales (doctrina de la neurona); y, **Charles Scott Sherrington**[141], un médico neurofisiólogo británico, premio Nobel de Medicina, que estudió las funciones de la corteza cerebral.

El comportamiento, pensamiento y emociones están mediatizados por una compleja red de neuronas que operan en el cerebro. Se estima que el cerebro tiene aproximadamente 10^{11} células nerviosas (neuronas) y unas 10^{14}, conexiones (sinapsis). Como resultado de esta impresionante cantidad de neuronas y conexiones, el cerebro es una estructura de procesamiento paralelo – distribuido altamente eficiente, a pesar que las neuronas son procesadores mucho más lentos que los circuitos integrados basados en el silicio. En un procesador de basado en silicio el tiempo de proceso está en el orden de los nanosegundos (10^{-9} segundos), mientras que en las neuronas los eventos se procesan en el orden de los milisegundos (10^{-3} segundos)[142].

En la segunda mitad del siglo XX, la disponibilidad de computadores digitales permitió a los científicos recrear y estudiar redes neuronales simuladas y procesadas en chips de silicio. El desarrollo de modelos de sistemas neuronales tanto formales como computacionales, ocurrido en los últimos 50 años, ha sido impulsado por investigaciones en neurociencias, ciencias cognitivas, neurociencia computacional e ingeniería neuronal.

9.1.1 Neurociencias

La neurociencia es una disciplina científica que estudia la psicología, función, estructura, desarrollo, genética, bioquímica, fisiología, farmacología, y la patología del sistema nervioso.

Tradicionalmente se consideraba como una rama de las ciencias biológicas. Sin embargo, recientemente ha habido una convergencia de intereses de otras disciplinas, incluyendo ciencias de computación, estadística, física, matemática, filosofía y medicina.

---

[140] http://es.wikipedia.org/wiki/Santiago_Ram%C3%B3n_y_Cajal
[141] http://es.wikipedia.org/wiki/Charles_Scott_Sherrington
[142] Nakamura K. *Neural Processing in the Subsecond Time Range in the Temporal Cortex*. Neural Computation, April 1, 1998, Vol. 10, No. 3, Pages 567-595

El ámbito de aplicación de la neurociencia se ha ampliado para incluir cualquier investigación científica sistemática experimental y teórica del sistema nervioso central y periférico de los organismos biológicos.

Las metodologías empleadas por los neuro-científicos se han expandido enormemente, desde el análisis bioquímico y genético de la dinámica de las células nerviosas individuales y sus componentes moleculares hasta las representaciones de la percepción y las habilidades motrices en el cerebro. La neurociencia está en la frontera de la investigación del cerebro y la mente.

El estudio del cerebro se está convirtiendo en la piedra angular para la comprensión de la forma en que percibimos e interactuamos con el mundo exterior y, en particular, cómo la experiencia humana y la biología humana se influencian una con otra.

Es probable que el estudio del cerebro se convierta en uno de los esfuerzos intelectuales centrales en las próximas décadas.

9.1.2 Ciencia Cognitiva

La **ciencia cognitiva**[143] es el estudio interdisciplinario de la mente y la inteligencia. Está relacionada con la filosofía, la psicología, la inteligencia artificial, la neurociencia, la lingüística y la antropología. Sus orígenes intelectuales están en la mitad de la década de 1950 cuando los investigadores en varios campos comenzaron a desarrollar teorías de la mente sobre la base de representaciones complejas y procedimientos de cálculo.

La hipótesis central de la ciencia cognitiva es que el pensamiento puede ser comprendido mejor en términos de estructuras representativas en la mente y procedimientos que operan sobre esas estructuras.

La mayoría de los trabajos en ciencias cognitivas asumen que la mente usa representaciones análogas a la estructuras de datos de los computadores; y, procedimientos similares a algoritmos computacionales.

Actualmente, la principal fuente de referencia es la Sociedad de Ciencia Cognitiva[144]. Se inicia como organización a mediados de la década de 1970 y comienza a publicar la revista *Cognitive Science*[145]. Desde entonces, más de sesenta universidades de Norteamérica, Europa, Asia y Australia han establecido programas de investigación y muchas otras han creado cursos en ciencia cognitiva.



Figura 48. Logo de la Cognitive Science Society

143 http://plato.stanford.edu/entries/cognitive-science/
144 http://cognitivesciencesociety.org/index.html
145 http://cognitivesciencesociety.org/journal_csj.html

9.1.3 Neurociencia Computacional

Tiene sus raíces en la modelación matemática de la dinámica de la membrana neuronal realizada por **Hodgkin y Huxley**[146] como un intento para comprender el funcionamiento de las neuronas cerebrales.

Entre los principales aspectos que trata la neurociencia computacional, están el tipo de comunicación usado por las neuronas, el efecto de las substancias químicas en el comportamiento neuronal, la dinámica de conjuntos neuronales y la capacidad teórica de computación neuronal.

9.1.4 Ingeniería Neuronal

Su inicio puede ser asociado a la descripción a nivel lógico de una neurona, propuesta por **McCullock y Pitts**[147], que permitió reproducir la funcionalidad de modelos de neuronas para realizar la ingeniería de máquinas inteligentes.

Los problemas tratados por la ingeniería neuronal incluyen: control robusto de sistemas robóticos, algoritmos de aprendizaje, arquitecturas de alto nivel capaces de reproducir habilidades cognitivas y la implementación de redes neuronales en software (SW) y hardware (HW).

En términos generales, las redes neuronales ofrecen las siguientes propiedades y capacidades que son muy útiles desde la perspectiva de la ingeniería neuronal[148]:

- No linealidad
- Mapeo Entrada – Salida
- Adaptabilidad
- Respuesta basada en evidencia
- Información contextual
- Tolerancia a fallas
- Posibilidad de implementación en HW, VLSI (*Very Large Scale Integration*)
- Uniformidad de análisis y diseño
- Analogía neuro-biológica

9.2 Fundamentos de las Redes Neuronales

El sistema nervioso humano y, en especial, el cerebro, está compuesto por un conjunto de células nerviosas, también llamadas neuronas. Una neurona es una célula altamente

---

[146] Hodgkin, A., and Huxley, A.: *A quantitative description of membrane current and its application to conduction and excitation in nerve*. *J. Physiol.* **117,** 1952 pp:500–544

[147] McCulloch, W. and Pitts, W. *A logical calculus of the ideas immanent in nervous activity*. Bulletin of Mathematical Biophysics, 7, 1943, pp:115 - 133.

[148] Haykin, S. *Neural Networks and Learning Machines, 3rd Ed*. Prentice Hall. USA, 2008.

especializada en el proceso de la información, cuya estructura básica puede observarse en la siguiente Figura.

La morfología de una neurona comprende tres elementos principales: el *soma*, o núcleo; las *dendritas*, o terminaciones de la neurona que actúan como contactos funcionales de entrada con otras neuronas; y el *axón* o eje, una rama más larga que será la encargada de conducir el impulso nervioso y que finaliza también en diversas ramificaciones.

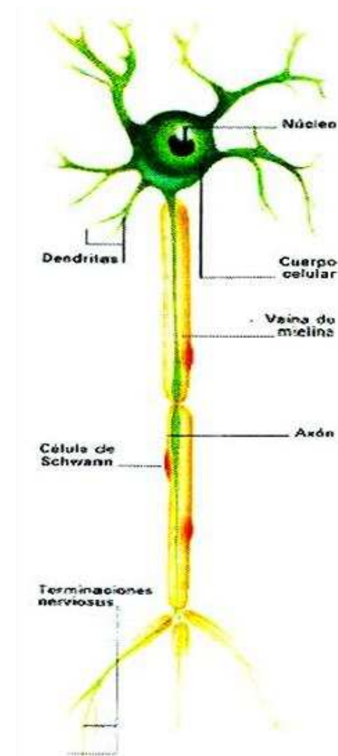


Figura 49. Estructura de una Neurona

La comunicación entre neuronas se realiza a través de las llamadas *sinapsis*, que son los puntos de conexión entre las fibras terminales del axón de una neurona y una dendrita de otra.

Las sinapsis también reciben el nombre de *saltos sinápticos*, ya que en la mayoría de los casos, fibras terminales y dendritas no están en contacto, sino separadas por una pequeña distancia. Por ello, al contrario de lo que mucha gente piensa, el impulso nervioso no es de naturaleza eléctrica, sino electroquímica.

El impulso nervioso producido por una neurona se propaga por el axón y al llegar al extremo, las fibras terminales pre-sinápticas liberan unos compuestos químicos llamados *neurotransmisores*.

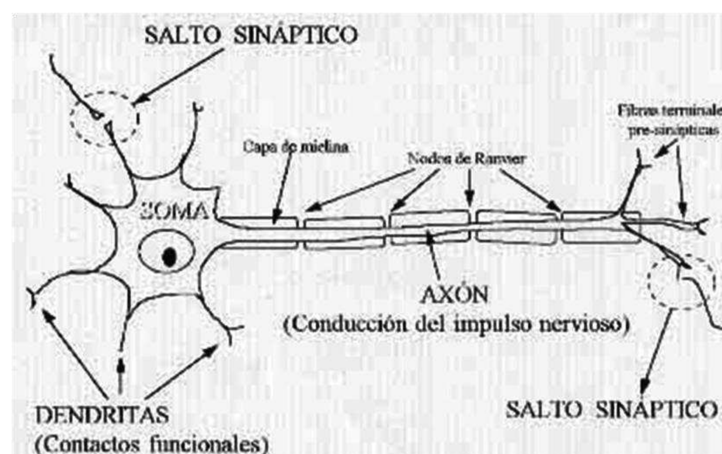


Figura 50. Comunicación entre Neuronas

Los neurotransmisores alteran el estado eléctrico de la membrana post-sináptica. En función del neurotransmisor liberado, el mecanismo puede resultar *excitador* o *inhibidor* para la neurona receptora.

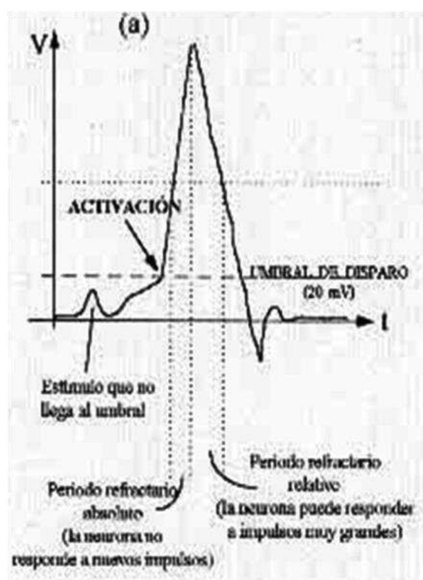


Figura 51. Respuesta de una Neurona Biológica

En el soma de una neurona se integran todos los estímulos recibidos a través de todas las dendritas. Si como resultado se supera un potencial de activación, la neurona se dispara generando un impulso que se transmitirá a través del axón.

Este impulso no es constante y la capacidad de reaccionar de la neurona varía con el tiempo hasta volver al estado de reposo. La comunicación tiene lugar a través de trenes de pulsos, por lo que los mensajes se encuentran modulados en frecuencia.

Tanto el salto electroquímico como la aparición de períodos refractarios, limitan mucho la velocidad de la neurona biológica y el rango de frecuencia de los mensajes, que oscila entre unos pocos y algunos cientos de hertzios (ciclos/segundo).

Por ello, el tiempo de respuesta se ve limitado al orden de milisegundos, mucho más lenta que un circuito electrónico.

9.3 Modelo Neuronal

El modelo de neurona artificial propuesto por **McCulloch** y **Pitts** consiste de una unidad con función de activación tipo escalón (binaria) similar a la que se observa en la Figura.

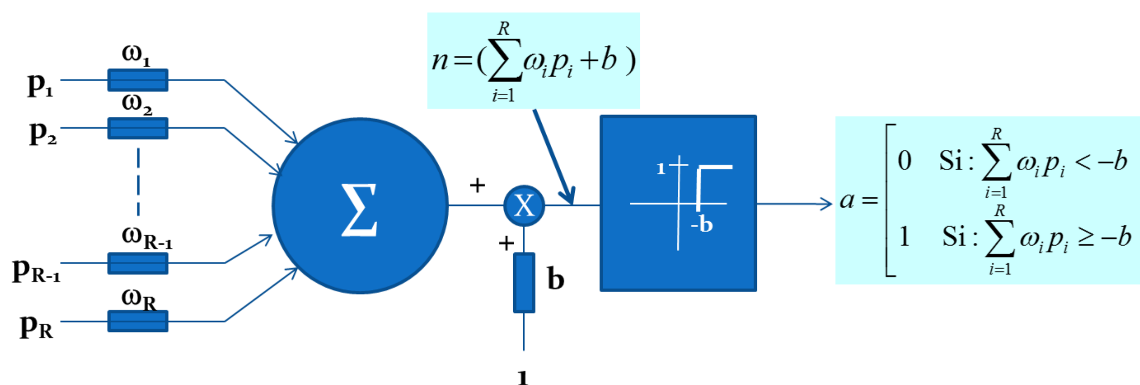


Figura 52. Modelo de una neurona propuesto por McCulloch y Pitts (1943).

Este modelo de neurona artificial presenta numerosas analogías con las neuronas biológicas:

- Los cables o conexiones son análogos a dendritas y axones,
- Los pesos de ponderación de las conexiones equivalen a las sinapsis; y,
- El umbral de activación de la neurona representa la actividad del soma.

Dependiendo de su signo, los pesos sinápticos pueden actuar tanto como activadores (signo positivo) o como inhibidores (signo negativo).

- La unidad realiza una función de proceso, sumando los productos de los valores de las entradas p_i por los pesos sinápticos ω_i:

$$\sum_{i=1}^{R} \omega_i \, p_i$$

- Si el resultado obtenido es igual o mayor al valor del umbral (-b), la neurona se activa produciendo como respuesta el *valor de activación* (a = 1).

$$\sum_{i=1}^{R} \omega_i \, p_i \geq -b$$

- Por el contrario, si el resultado de aplicar la función de proceso a los valores de entrada no superara el valor del *umbral* (-b), la neurona permanecerá inactiva y su salida será nula (a = 0).

$$\sum_{i=1}^{R} \omega_i \, p_i < -b$$

Desde la publicación del trabajo de McCulloch y Pitts se han dado numerosas generalizaciones a su modelo de neurona, pero la más empleada consiste en la sustitución de la función de activación o de transferencia tipo escalón, por una función rampa, por una Gaussiana o por una sigmoide:

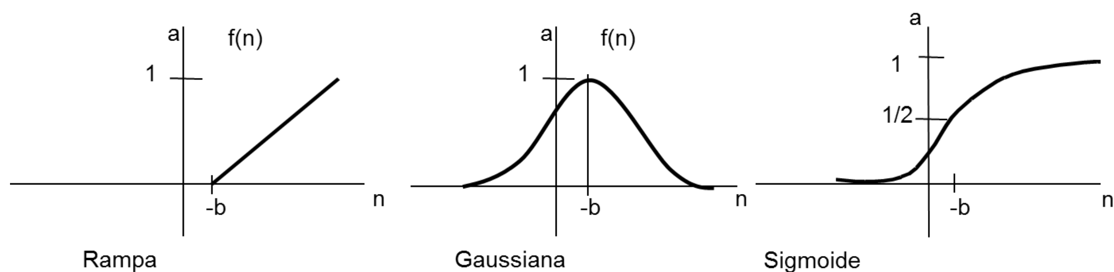


Figura 53. Funciones de activación mono polares

Las funciones de activación pueden ser de tipo mono polar como las indicadas en la figura anterior o también pueden ser bipolares, como se indica a continuación.

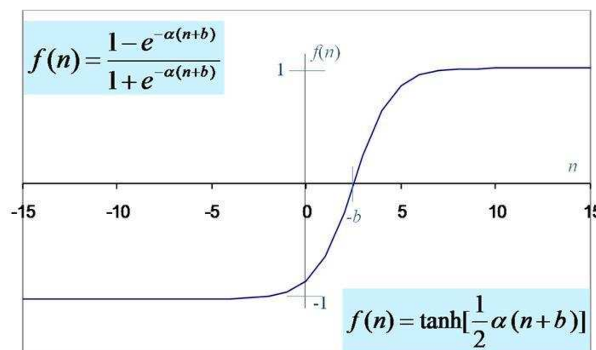


Figura 54. Función de Activación Sigmoide Bipolar

En la práctica, para simplificar la notación y la programación del modelo, se suele incluir el umbral de activación dentro del sumatorio. El operador "asterisco" (*) representa el producto escalar de los vectores de las señales de entrada y de los pesos de las conexiones sinápticas.

$$a = f\left(\sum_{i=1}^{R} \omega_i p_i + b\right) = f(\mathbf{W} * \mathbf{p} + b)$$

$$\text{Donde} \begin{cases} \mathbf{W} \text{ es la Matriz de Pesos} \\ \mathbf{p} \text{ el Vector de Atributos} \\ b \text{ el Umbral de Decisión} \end{cases}$$

$$\text{Si}: p_0 = 1 \text{ y } \omega_0 = b$$

$$a = f\left(\sum_{i=0}^{R} \omega_i p_i\right) = f(\mathbf{W} * \mathbf{p_a})$$

$$\mathbf{p_a} = \text{Vector de atributos aumentado}$$

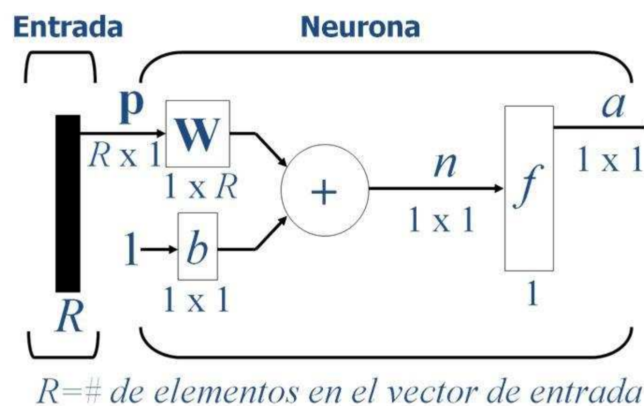


Figura 55. Modelo Abreviado de una Neurona con Vector a la Entrada

9.4 Aprendizaje de una Neurona Artificial

El psicólogo **Donald Hebb**, en sus observaciones de experimentos neurobiológicos dedujo que si las neuronas de ambos lados de una sinapsis se activan simultáneamente repetidas veces, el valor de la sinapsis se incrementa. Este principio es lo que se conoce como postulado de **Aprendizaje Hebbiano (*Hebbian Learning*)**[149,150].

El aprendizaje de una neurona y de una red neuronal se traduce en la adaptación de los pesos de sus conexiones sinápticas, para disminuir el error entre la salida producida ante una entrada y la salida correcta. El proceso de aprendizaje implica la siguiente secuencia de eventos:

- La red neuronal es estimulada por muestras de su entorno.
- La red neuronal sufre cambios como resultado del estímulo aplicado.
- La red neuronal responde al entorno demostrando el entrenamiento recibido, debido a los cambios que han ocurrido en su estructura interna.

---

[149] Hebb, D. O. *The Organization of Behavior: A neuropsychological theory*. New York: Wiley. 1949.
[150] http://en.wikibooks.org/wiki/Artificial_Neural_Networks/Hebbian_Learning

9.4.1 Entorno y Conocimiento

No existe una teoría bien desarrollada para optimizar la arquitectura de una red neuronal requerida para interactuar con un cierto entorno de interés y para evaluar la forma en que los cambios en la arquitectura de la red afectan la representación del conocimiento dentro de la misma.

Respuestas satisfactorias a estos problemas se pueden encontrar a través de exhaustivos estudios experimentales, con el diseñador de la red neuronal convertido en parte esencial de la estructura del ciclo de aprendizaje. En este contexto:

> ***El conocimiento se refiere a Información almacenada o a modelos utilizados por personas o máquinas para interpretar, predecir y responder apropiadamente en un entorno determinado[151].***

9.4.2 Representación del Conocimiento

La representación del conocimiento, involucra dos aspectos: Qué información se tiene que hacer explícita; y, cómo se la debe codificar para su uso subsiguiente.

Por otra parte, el conocimiento acerca del entorno de interés, comprende dos tipos de información:

- Lo que se conoce acerca del estado actual y cómo fue antes (conocimiento previo); y,
- Las observaciones del ambiente donde va a operar la red neuronal (mediciones), realizadas por medio de sensores diseñados para obtener atributos de ese entorno.

Usualmente, estas observaciones son inherentemente ruidosas y propensas a errores debidos a imperfecciones de los sensores y de los sistemas de medición. De todas maneras, las observaciones así obtenidas son representaciones simbólicas que pueden ser suficientemente útiles como para conformar una fuente de información, de donde se puede seleccionar conjuntos de ejemplos para:

- Entrenar una red neuronal artificial (***Training Set***).
- Comprobar el aprendizaje de una red neuronal artificial (***Testing Set***).
- Verificar la operación de una red neuronal artificial (***Working Set***).

9.4.3 Objetos de Interés y Patrones

Los objetos que constituyen el entorno de interés pueden ser datos, caracteres impresos, señales audibles, visuales o electrónicas, imágenes, estados de un sistema o cualquier cosa susceptible de ser representada, aprendida, reconocida o clasificada. Se los representa mediante patrones.

---

[151] Fischler M. A., Firschein O. *Intelligence: The Eye, the Brain and the Computer*. Addison-Wesley, 1987.

Un patrón es una abstracción de los atributos de un objeto de interés. Sintetiza sus principales rasgos característicos. Un patrón usualmente se lo representa por un vector n-dimensional que integra un conjunto de mediciones obtenidas de los atributos del objeto de interés o de sus interrelaciones.

La representación del conocimiento mediante patrones y la selección del correspondiente conjunto de vectores constituyen las principales tareas del proceso de diseño de soluciones con redes neuronales, y son la clave de su operación.

9.4.4 Entrenamiento y Aprendizaje Formal

El conjunto de ejemplos, representa el conocimiento del entorno de interés que la red neuronal debe aprender a través del entrenamiento. Cada ejemplo del conjunto de entrenamiento puede ser etiquetado o no. Un ejemplo etiquetado consiste de un par entrada – salida, esto es, un vector de entrada emparejado con su correspondiente respuesta deseada. Estos pueden ser costosos de obtener ya que necesitan de un experto para proveer a cada patrón de ejemplo, la respuesta deseada.

Por otra parte, los ejemplos no etiquetados corresponden a instancias de la señal de entrada a la red, que no requieren de profesor para determinar la salida deseada.

Dado un conjunto de ejemplos, etiquetados o no, la complejidad de aprender un concepto es una función de tres factores: la tolerancia al error (e), el número de rasgos presentes en los ejemplos (n) y el tamaño de la regla necesaria para discriminar (t).

Un sistema es capaz de aprender un concepto si, dado un conjunto de ejemplos positivos y negativos, puede producir un algoritmo que pueda clasificar correctamente en el futuro, con probabilidad inversamente proporcional a la tolerancia especificada para el error.

Si el número de ejemplos requeridos para el entrenamiento de un sistema está dado por un polinomio $p(e,n,t)$, entonces se dice que el concepto es susceptible de ser aprendido. (**Teoría del Aprendizaje Formal**[152])

9.4.5 Tamaño de la Muestra

Para los procesos de aprendizaje, una investigación realizada por **Desmond H. Foley**[153] sugiere que, para minimizar el error del sistema, el número de ejemplos para cada clase, elegidos aleatoriamente para conformar el conjunto de entrenamiento debe ser al menos tres veces el número de rasgos utilizados para representar cada caso, esto es igual o mayor a **tres veces** la dimensión del vector del patrón característico.

---

[152] http://plato.stanford.edu/entries/learning-formal/

[153] Foley, D. H. *Considerations of Sample and Feature Size*. IEEE Transactions on Information Theory, Vol IT-18, 1972, pp. 618.

9.4.6 Preprocesamiento

El entrenamiento de una red neuronal puede realizarse en forma más eficiente si se aplican ciertos pasos de preprocesamiento a los vectores de entrada y salidas esperadas. Varios pueden ser los métodos a aplicarse:

- Análisis del poder discriminante de los atributos.
- Escalamiento de entradas y salidas (rango entre -1 y +1)
- Normalización de entradas y salidas (media cero y desviación estándar unitaria)
- Análisis de componentes principales (reducción dimensional)

9.4.7 Poder Discriminante de Atributos

La evaluación del poder discriminante y la selección de atributos es una etapa importante en el diseño de un sistema de reconocimiento de patrones.

La prueba de **William Kruskal** y **W. Allen Wallis**[154] es un método no paramétrico que se utiliza para probar si dos o más grupos de datos provienen de la misma población. Es idéntico al método paramétrico **ANOVA** en el que se asume normalidad de los datos.

9.4.8 Estilos de Entrenamiento

Para el entrenamiento, la presentación de ejemplos a la red y el correspondiente ajuste a los valores de las conexiones sinápticas, puede hacerse utilizando dos estrategias:

- **Entrenamiento Incremental**.- Los pesos de las conexiones sinápticas y los valores de los umbrales de decisión, son actualizados cada vez que un vector de entrada es presentado a la red.
- **Entrenamiento por Lotes**.- Los pesos de las conexiones sinápticas y los valores de los umbrales de decisión, son actualizados después de que todo el grupo de vectores de entrada han sido presentados a la red (**época**).

9.4.9 Generalización

Uno de los problemas que ocurren durante el entrenamiento de redes neuronales es el sobre entrenamiento.

El error en el conjunto de entrenamiento alcanza un valor muy pequeño, pero cuando nuevos vectores son presentados a la red el error es grande.

La red ha memorizado los ejemplos de entrenamiento, pero no tiene capacidad de generalización a situaciones nuevas.

9.4.10 Matriz de Confusión

Describe los errores en los que incurre una red neuronal al tratar de clasificar un conjunto de vectores representativos del problema para el que fue entrenada.

---

[154] http://udel.edu/~mcdonald/statkruskalwallis.html

Los grupos de clasificación componen tanto las filas como las columnas. Los valores en su diagonal representan los casos correctamente clasificados y aquellos fuera de la diagonal los errores cometidos.

Grupos o Clases		1	2	3	4	5
Grupos o Clases	1	c_{11}	e_{12}	e_{13}	e_{14}	e_{15}
	2	e_{21}	c_{22}	e_{23}	e_{24}	e_{25}
	3	e_{31}	e_{32}	c_{33}	e_{34}	e_{35}
	4	e_{41}	e_{42}	e_{43}	c_{44}	e_{45}
	5	e_{51}	e_{52}	e_{53}	e_{54}	c_{55}

Figura 56. Matriz de Confusión

9.4.11 Tipos de Errores

A fin de evaluar el desempeño de una red neuronal, una vez concluido el proceso de entrenamiento, se pueden definir dos tipos de errores:

- **Error Optimista**.- Se obtiene cuando se calcula el error de una matriz de confusión construida utilizando el conjunto de entrenamiento como conjunto de prueba.
- **Error Pesimista**.- Resulta del cálculo del error dado por una matriz de confusión construida utilizando un conjunto de prueba diferente al conjunto de entrenamiento.

En algún punto de la región comprendida entre el lugar geométrico de estas dos funciones de error, radica el **error real** asociado a la operación de la red neuronal.

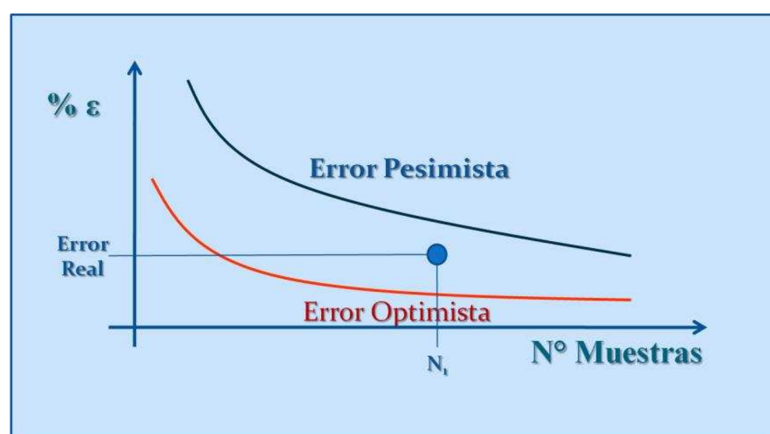


Figura 57. Estimación de error de desempeño en Redes Neuronales

9.5 Proceso de Aprendizaje

Una de las propiedades más interesantes de una red neuronal es su habilidad para aprender de su entorno y a través del aprendizaje ir mejorando su desempeño a lo largo del tiempo de acuerdo a alguna medida prescrita.

Simon Haykin[155], plantea que en el proceso de aprendizaje se puede distinguir dos componentes: Los **algoritmos o reglas de aprendizaje**; y, la forma cómo una red neuronal se relaciona con su entorno para modelar el entorno en el que tiene que operar, en este caso se habla de **paradigmas de aprendizaje**.

Con respecto a los paradigmas de aprendizaje, al igual que en el caso de los seres humanos, una red neuronal puede aprender con o sin la ayuda de un profesor. Al aprendizaje que se realiza con el apoyo de un profesor, se lo puede denominar **Aprendizaje Supervisado**, mientras que al aprendizaje que se lo realiza sin profesor se lo puede dividir en **Aprendizaje por Refuerzo** y en **Aprendizaje No-Supervisado**.

9.5.1 Aprendizaje Supervisado

En términos conceptuales, el paradigma de aprendizaje supervisado requiere de un profesor que tenga conocimiento del entorno. El configura el conjunto de vectores descriptivos que representan ese conocimiento. En este caso la red ve el entorno a través de la codificación realizada por el profesor. En virtud de su conocimiento, el profesor es capaz de proveer a la red neuronal la respuesta deseada correspondiente a cada vector utilizado para el entrenamiento. Los parámetros de la red son ajustados bajo la acción combinada de los vectores de entrada y las señales de error. Las señales de error corresponden a la diferencia entre la respuesta deseada y la respuesta actual de la red.

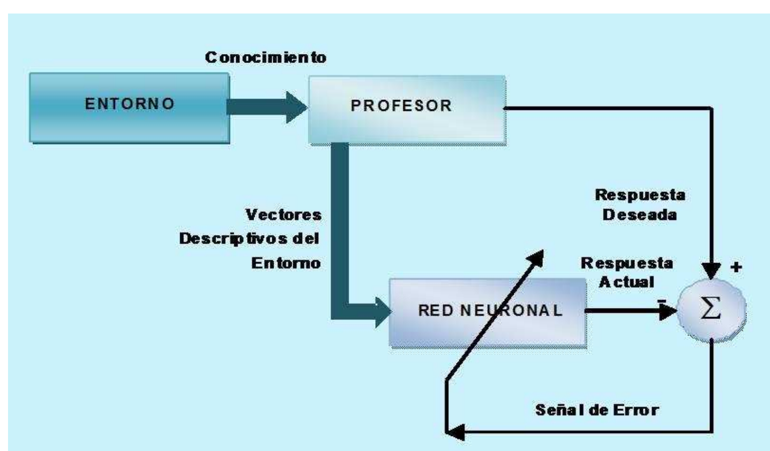


Figura 58. Modelo de Aprendizaje con Profesor[156].

Los ajustes a los parámetros de la red se realizan en forma iterativa, paso a paso, tratando de que eventualmente el desempeño de la red emule el desempeño del profesor. La

---

[155] Haykin, S. *Neural Networks-A Comprehensive Foundation, 1st Ed.* Macmillan Publishig Co. USA, 1994.

[156] Haykin, S. *Neural Networks and Learning Machines, 3rd Ed.* Prentice Hall. USA, 2008.

emulación se asume óptima utilizando algún criterio estadístico. De esta manera, el conocimiento del profesor acerca del entorno, es transferido a la red neuronal, como código sub-simbólico y almacenado en forma de pesos sinápticos, representando una memoria de largo plazo. Entonces la red está lista para actuar por sí misma.

9.5.2 Aprendizaje por Refuerzo

En el aprendizaje por refuerzo, el aprendizaje del mapeo entrada – salida, se realiza a través de interacción continua con el entorno, para minimizar un índice de desempeño escalar. En el siguiente modelo, se muestra que el aprendizaje por refuerzo se centra alrededor de un crítico, encargado de convertir una señal de refuerzo primario recibida desde el entorno en una señal de alta calidad de refuerzo heurístico. Ambas señales son de tipo escalar.

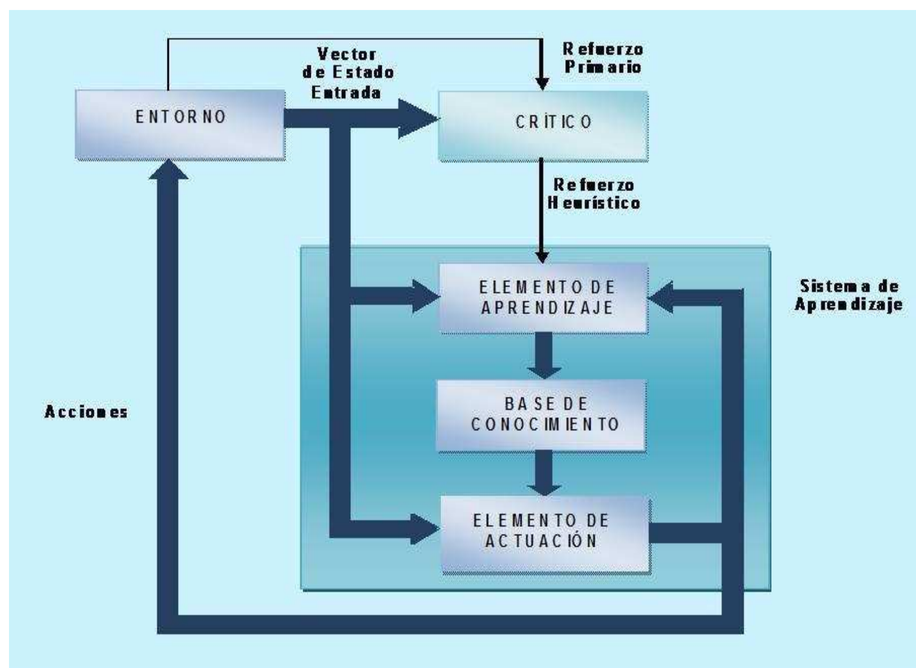


Figura 59. Modelo de Aprendizaje por Refuerzo

Este procedimiento permite al sistema de aprendizaje interactuar con el entorno y desarrollar la habilidad de desempeñar una tarea prescrita, basado en los resultados de estas interacciones.

9.5.3 Aprendizaje No - Supervisado

En el aprendizaje no supervisado o auto – organizativo, no existe un profesor o un crítico para supervisar el proceso de aprendizaje. No hay ejemplos específicos de la función a ser aprendida por la red. A cambio se provee una medida, independiente de la tarea, que determina la calidad de la representación que debe ser aprendida por la red, y sus parámetros libres son optimizados con respecto a esa medida.

Una vez que la red se ha sintonizado con las regularidades estadísticas de los datos de entrada, desarrolla la habilidad de formar representaciones internas para los rasgos característicos de la entrada y por lo tanto crea nuevas clases automáticamente.



Figura 60. Modelo de Aprendizaje No-Supervisado

Para lograr el aprendizaje no – supervisado, se puede utilizar una regla de aprendizaje competitivo. El sistema de aprendizaje está conformado por una capa de entrada que recibe los datos disponibles y una capa en la que las neuronas compiten unas con otras por la oportunidad de responder a los atributos contenidos en los datos de entrada. De acuerdo con la regla de aprendizaje competitivo, la neurona con mayor cantidad de entradas gana y se activa para un determinado atributo, mientras que las otras se desactivan.

9.6 Tareas Básicas de Aprendizaje

La decisión acerca de una arquitectura de red y de un algoritmo de aprendizaje, están influenciadas por la tareas de aprendizaje, cuya naturaleza diversa atestigua la universalidad de la redes neuronales.

9.6.1 Asociación de Patrones

El aprendizaje asociativo es una de las tareas más comunes en la memoria humana. Pueden distinguirse dos tipos: autoasociación y heteroasociación.

- En **autoasociación**, se requiere que la red almacene un conjunto de patrones (vectores) mediante la presentación repetitiva de ellos durante el proceso de aprendizaje, usualmente de tipo no-supervisado. Posteriormente, al presentarse una versión incompleta o ruidosa del patrón almacenado, la red es capaz de recuperar y entregar en su salida el patrón completo.

> ***Para una operación confiable de la red autoasociativa, el número de patrones aleatorios almacenados (p), debe ser menor que el 14% del total de neuronas (N) de la memoria auto-asociativa[157]:***
>
> $$p < 0,14.N$$

[157] Müller B and Reinhardt J. *Neural Networks - An Introduction*. Physics of Neural Networks Series. Springer-Verlag, Germany, 1991.

- En **heteroasociación**, se trata de que la red asocie un conjunto arbitrario de patrones de entrada, con otro conjunto arbitrario de patrones de salida. En este caso el aprendizaje es de tipo supervisado.

9.6.2 Reconocimiento de Patrones

El reconocimiento de patrones es un proceso en el cual un patrón recibido es asignado a uno de un número prescrito de clases. El reconocimiento de patrones es un proceso de naturaleza estadística, en el que los patrones son representados por puntos en un espacio de decisión `n-dimensional`, cuyos ejes son los `n` atributos codificados en los vectores de entrada. El espacio de decisión está dividido en regiones, cada región está asociada a una clase. Las fronteras de decisión se determinan durante el proceso de entrenamiento.

9.6.3 Aproximación de Funciones

El problema de aproximación de funciones puede ser visto como un tipo de aprendizaje supervisado, en el que la habilidad de una red puede ser explotada para:

- Identificación de sistemas; y,
- Modelación inversa.

9.6.4 Control

El control de una planta es otra de las tareas que una red es capaz de realizar. En este contexto, se entiende por planta a un proceso o parte crítica de un sistema, que debe ser mantenido en condición controlada.

Para poder controlar la planta, es necesario conocer sus parámetros. Pero usualmente, estos son variables y dependen del punto de operación, por lo que la red neuronal constituye un sistema idóneo para su determinación dinámica, ya sea por aprendizaje directo[158], o por aprendizaje indirecto[159].

9.6.5 Formación de Haz

Se utiliza para distinguir entre las propiedades espaciales de una señal objetivo y el ruido de fondo. La formación de Haz se utiliza en sistemas de radar y sonar, donde la tarea primaria es detectar y seguir un objetivo de interés en la presencia combinada de ruido de recepción y señales de interferencia. Esta tarea tiene dos elementos que la complican:

- La señal objetivo se origina en una dirección desconocida; y,
- No existe información previa disponible acerca de las señales que interfieren.

---

[158] Widrow B. and Wallach E. *Adaptive Inverse Control*, Prentice Hall, 1996

[159] W.H. Schiffman and H.W. Geffers, *Adaptive control of dynamic systems by backpropagation networks*. Neural Networks 6 (1993), pp. 517–524

9.7 Perceptrones

En 1958, **Frank Rosenblatt** desarrolló una aplicación práctica para la neurona de **McCulloch** y **Pitts**, utilizando las ideas de aprendizaje de **Hebb**. Construyó una máquina capaz de reconocer objetos de forma visual. Así nació el **perceptrón**, la primera red neuronal de la historia[160].

El diseño del perceptrón consistía en una capa de elementos sensores, cuyas salidas se conectaban a las entradas de una neurona de **McCulloch** y **Pitts** a través de detectores de umbral de las señales de los sensores. El número de detectores de umbral era inferior al de elementos sensores, por lo que un detector podía conectarse a más de un sensor. La distribución de estas conexiones era aleatoria.

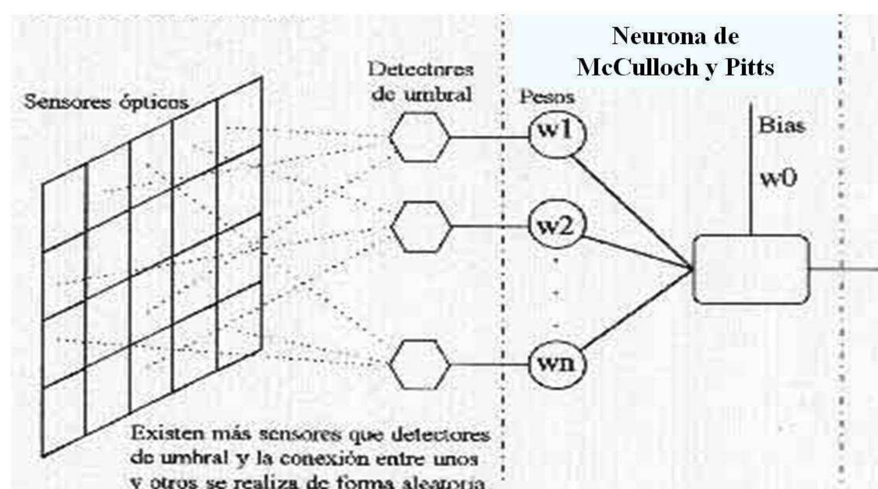


Figura 61. El Perceptrón de Frank Rosenblatt

Para 1962, Rosenblatt publicó un texto que contenía los resultados de sus investigaciones y mucho del material que utilizó por varios años para sus clases de la asignatura *Theory of Brain Mechanisms*[161]. Las expectativas que logró crear, más tarde fueron opacadas cuando en 1969 **Marvin Minsky** y **Seymour Papert** publicaron su libro *Perceptrons*[162]. Las demostraciones matemáticas, predicciones y limitaciones identificadas para los perceptrones dieron origen a un controversial cambio en la dirección de investigación en el campo de la inteligencia artificial. Por más de una década, se concentró en sistemas de procesamiento simbólico hasta que, en los años 1980, nuevas investigaciones demostraron que las prognosis realizadas por el libro, estuvieron erradas[163,164].

---

[160] Rosenblatt, F. *The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain*, Cornell Aeronautical Laboratory, Psychological Review, v65, No. 6, 1958, pp. 386–408.

[161] Rosenblatt, F. *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Spartan Books, USA, 1962.

[162] Minsky, M and Papert, S. *Perceptrons: An Introduction to Computational Geometry*. MIT Press, 1969.

[163] Rumelhart, D.E., J.L. McClelland and the PDP Research Group. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations*, Cambridge, MA: MIT Press, 1986.

[164] McClelland, J.L., D.E. Rumelhart and the PDP Research Group (1986). Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 2: Psychological and Biological Models, Cambridge, MA: MIT Press

9.7.1 Perceptrón Simple

El modelo de un perceptrón simple puede tener `R` entradas a una neurona con función de transferencia binaria que puede producir una salida (*a*) ya sea igual a *1* o igual a *0*. La entrada p_0 que siempre es igual a *1*, multiplicado por el peso *b*, constituye una variable que facilita el entrenamiento, sesgando horizontalmente la función de transferencia. Esto es, la transición de *0* a *1* ocurre en el punto del eje horizontal en el que el resultado del producto escalar del vector de pesos *W* por el vector de patrones *p* es igual a *−b* (**umbral de decisión**)

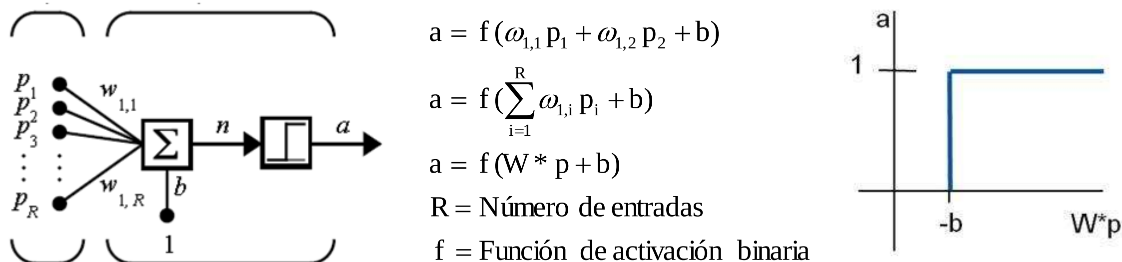


Figura 62. Modelo de un perceptrón simple según MATLAB[165]

El perceptrón simple tiene la habilidad de clasificar los vectores de entrada, dividiendo el espacio de atributos (*feature space*) en dos regiones delimitadas por la línea dada por:

$$W * p + b = 0$$

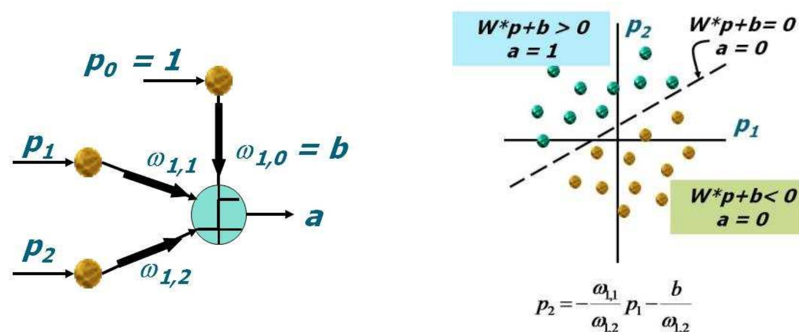


Figura 63. El perceptrón como clasificador lineal

9.7.2 Arquitectura de Perceptrones

Una red de perceptrones se configura como una capa de *S* neuronas conectadas a *R* entradas a través de un conjunto de pesos $w_{i,j}$. Los subíndices indican que el peso *w* conecta la entrada *j* a la neurona *i*.

La regla de aprendizaje del perceptrón, sólo puede entrenar una sola capa de neuronas. Las redes de perceptrones tienen varias limitaciones, por ejemplo:

- Las salidas de las neuronas sólo pueden tener valores de *0* o *1*.
- Los perceptrones sólo pueden clasificar conjuntos de vectores linealmente separables.

---

165 http://www.mathworks.com/

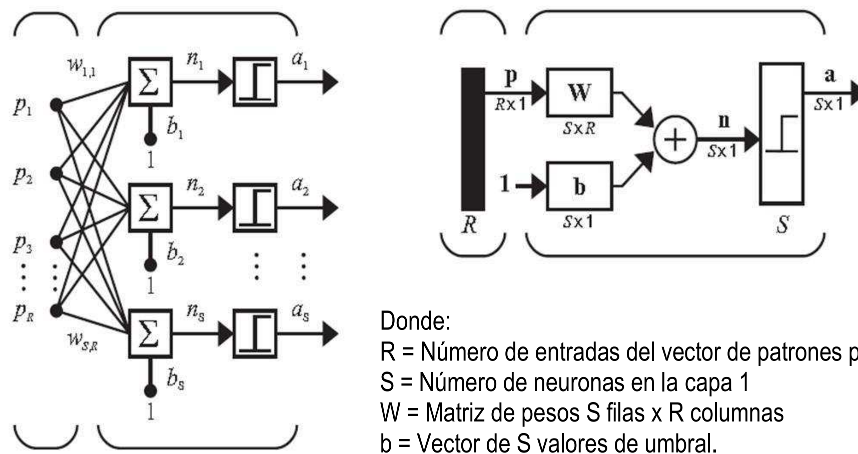


Figura 64. Arquitectura de Perceptrones en notación MATLAB

9.7.3 Regla de Aprendizaje del Perceptrón

Al perceptrón se lo entrena con ejemplos de comportamiento deseado, mediante la presentación de una secuencia de Q pares entrada (p) – salida (t) (Aprendizaje supervisado).

$$p^1t^1, p^2t^2, \ldots, p^Qt^Q$$

El objetivo es ir secuencialmente reduciendo el error (e), definido por la diferencia entre el vector de salida deseada (t) y las respuestas a la salida de las neuronas (a): $t - a$. La regla de aprendizaje del perceptrón se la puede resumir de la siguiente manera:

1. Inicializar todos los pesos W y umbrales de decisión b a valores aleatorios.
2. Para $q = 1$ hasta $q = Q$:
   a. Presentar un par entrada (p^q) – salida (t^q).
   b. Calcular la respuesta de las neuronas a^q al patrón p^q:

   $$a^q = W * p^q + b$$

   c. Calcular las matrices y realizar los ajustes a los pesos y a los valores del umbral de decisión:

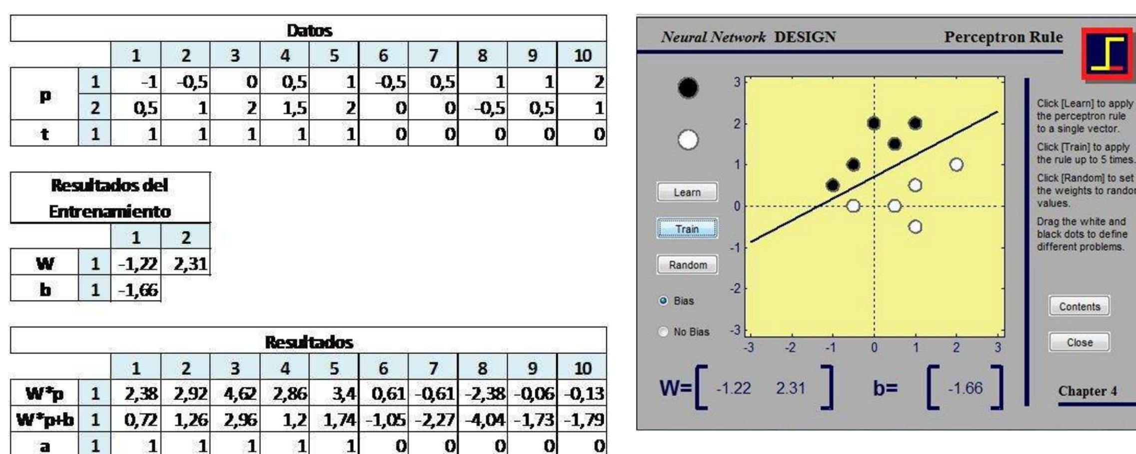
   $$\Delta W = e.(p^q)^T$$
   $$\Delta b = e = t^q - a^q$$

3. Si el error es mayor al aceptable, volver al paso 2.

El perceptrón aprende mediante entrenamiento incremental. Es decir las correcciones a los pesos y valores de umbral se realizan cada vez que se presenta un patrón a la entrada. La regla de aprendizaje del perceptrón ha demostrado que converge a una solución en un número finito de iteraciones, si los vectores de entrada son linealmente separables.

9.7.4 Ejemplo en MATLAB

En el paquete de MATLAB *Neural Network Toolbox Demos*, se encuentra un ejemplo (*Perceptron Learning Rule*) con el cual el usuario puede interactuar posicionando puntos blancos y negros de tal forma que se los pueda separar con una línea. Luego selecciona la opción con umbral de decisión (***bias***) y selecciona valores aleatorios para pesos y umbral de decisión. Puede comprobar que utilizando el entrenamiento incremental (***Learn***) la red eventualmente encuentra una solución para separar los grupos con la línea negra, mientras que si usa la opción de entrenamiento por lotes (***Train***), difícilmente la red puede encontrar solución. A través de la interfaz gráfica, el demo entrega los valores alcanzados por las variables durante el proceso de entrenamiento.



Datos		1	2	3	4	5	6	7	8	9	10
p	1	-1	-0,5	0	0,5	1	-0,5	0,5	1	1	2
	2	0,5	1	2	1,5	2	0	0	-0,5	0,5	1
t	1	1	1	1	1	1	0	0	0	0	0

Resultados del Entrenamiento		1	2
W	1	-1,22	2,31
b	1	-1,66	

Resultados		1	2	3	4	5	6	7	8	9	10
W*p	1	2,38	2,92	4,62	2,86	3,4	0,61	-0,61	-2,38	-0,06	-0,13
W*p+b	1	0,72	1,26	2,96	1,2	1,74	-1,05	-2,27	-4,04	-1,73	-1,79
a	1	1	1	1	1	1	0	0	0	0	0

Figura 65. Ejemplo de Perceptrón como clasificador lineal

En la figura del ejemplo, se posicionaron 5 puntos negros (`t = 1`) y cinco puntos blancos (`t = 0`). Las coordenadas se muestran en la matriz de datos `p`, con su respectivo vector de salidas deseadas `t`. En los resultados del entrenamiento logrado luego de presionar `Random` y después sucesivamente el botón `Learn` se muestra la matriz de pesos `W` y el valor alcanzado por el umbral de decisión `b`. Finalmente, en las matrices de resultados, utilizando las funciones del EXCEL, se comprueba la correcta respuesta de la red entrenada.

El estudiante interesado puede, dentro del entorno de trabajo de **MATLAB,** activar **Help** y en su tabla de contenido seleccionar **Neural Networks Toolbox ► Demos ► Perceptrons** y acceder a otros ejemplos, incluyendo el código fuente de los mismos.

9.8 Arquitecturas Neuronales

Una Red Neuronal, se define como un sistema procesador paralelo - distribuido, masivamente interconectado, capaz de almacenar y utilizar conocimiento experimental. Está diseñado para modelar la forma en que el cerebro realiza una tarea o función de interés. El conocimiento es adquirido mediante un proceso de adaptación y se almacena en las conexiones entre las neuronas de la red (sinapsis).

Existen diversas arquitecturas de Redes Neuronales Artificiales, según la forma cómo se interconectan las neuronas y cómo fluyen las señales dentro de la red:

- Estructuras Monocapa
- Estructuras Multicapa
- Estructuras en Malla
- Estructuras con conexiones hacia adelante (feedforward)
- Estructuras con conexiones hacia adelante y hacia atrás (recurrentes)

A continuación se presentan algunos ejemplos de arquitecturas neuronales:

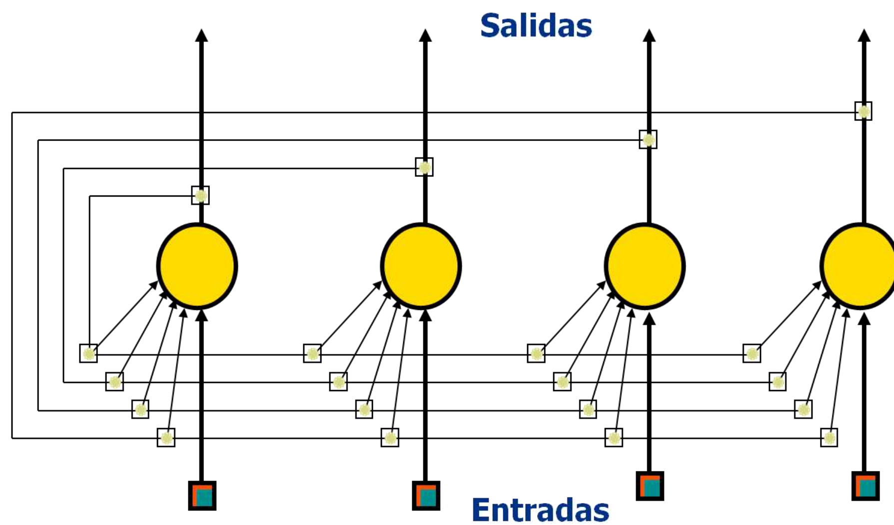


Figura 66. Estructura Mono Capa con conexiones recurrentes laterales

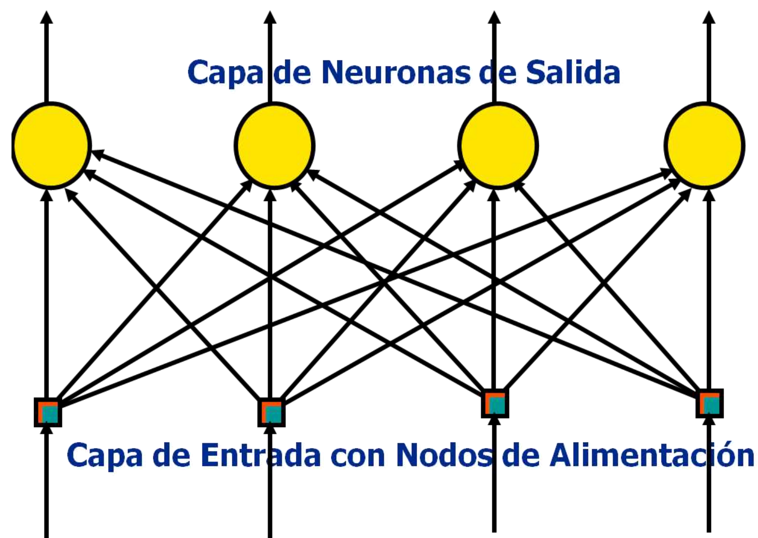


Figura 67. Estructura Multicapa sin Capa Escondida (2 Capas) con conexiones hacia adelante

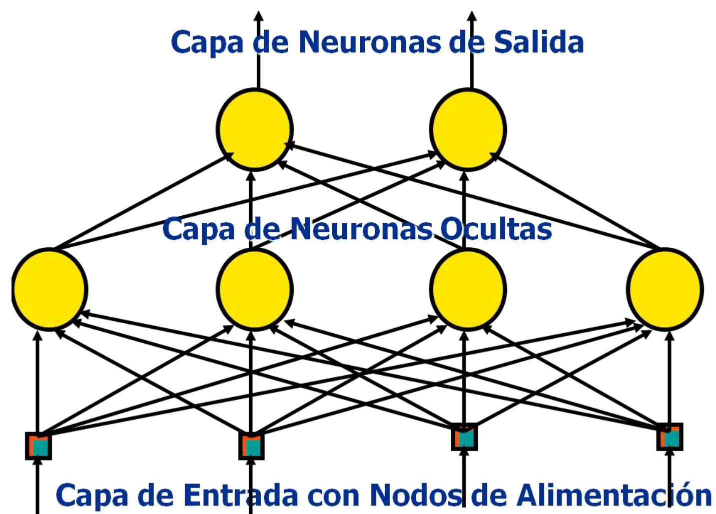


Figura 68. Estructura Multicapa con Capa Escondida (3 Capas) con conexiones hacia adelante

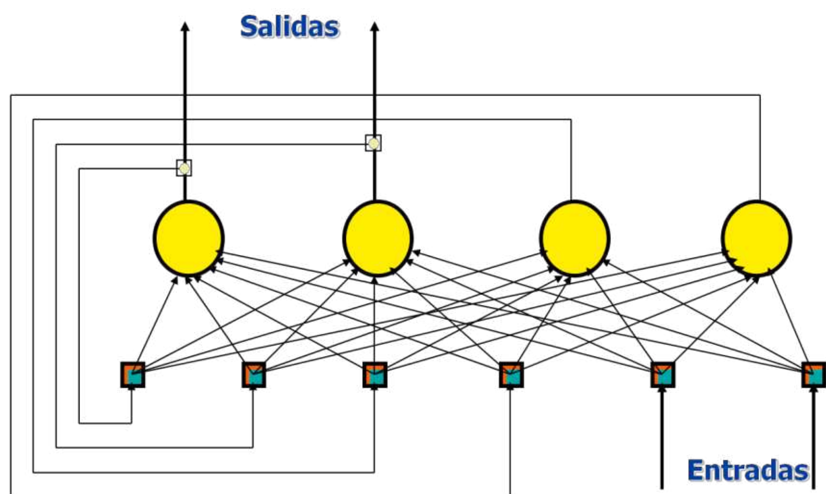


Figura 69. Estructura Multicapa con Neuronas Ocultas (3 Capas) con conexiones recurrentes

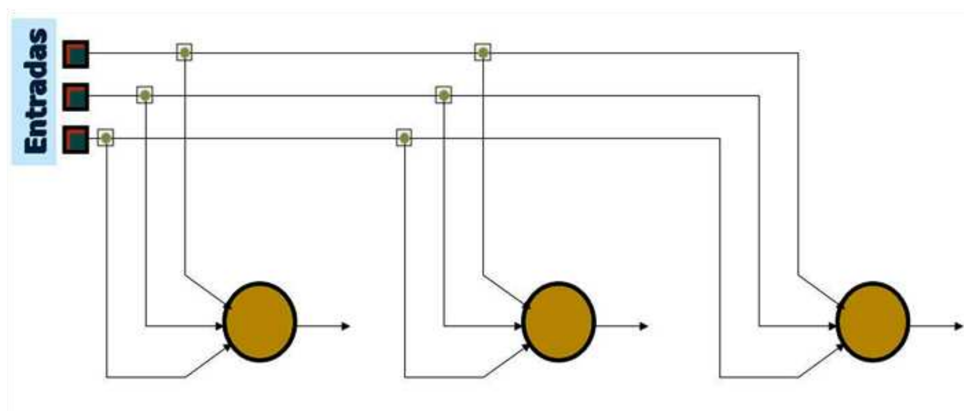


Figura 70. Estructura en Malla Unidimensional

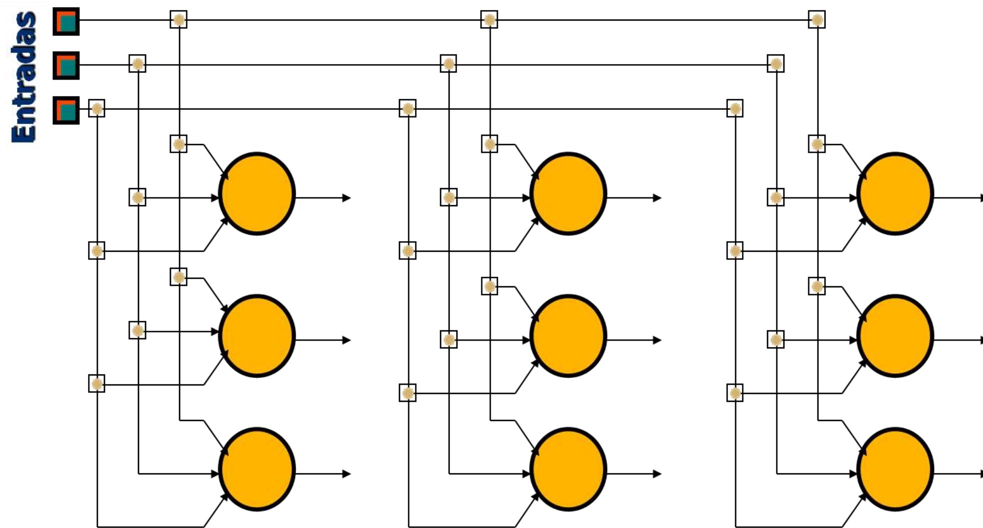


Figura 71. Estructura en Malla Bidimensional

9.9 Redes Neuronales de Aprendizaje Supervisado

El diseño de soluciones con Redes Neuronales Artificiales, comprende los siguientes pasos:

- Seleccionar una arquitectura de red neuronal, apropiada para resolver el problema identificado.
- Escoger un conjunto de ejemplos para entrenar la red, utilizando un algoritmo apropiado (Fase de entrenamiento o aprendizaje de la red).
- Probar la capacidad de la red entrenada para resolver el problema propuesto, utilizando un conjunto de datos de prueba (Fase de prueba de la red).
- Evaluar la operación de la red con un conjunto de datos de verificación, que no hayan sido utilizados en el entrenamiento o las pruebas (Fase de generalización).

9.9.1 Red Hopfield

La red **Hopfield** se la considera como una memoria asociativa o memoria direccionable según el contenido. Básicamente, es utilizada para almacenar de manera estable uno o más vectores objetivo[166]. Estos patrones pueden ser luego recuperados en respuesta a vectores similares, incompletos o contaminados con ruido, que se presenten en su entrada. Una memoria direccionable según el contenido es capaz de corregir errores en el sentido que puede superar información inconsistente en las claves presentadas.

El modelo de la red de Hopfield utiliza una función de transferencia bipolar lineal saturada. Esto es, para entradas menores a -1 la salida es -1. Para entradas en el rango -1 a +1, entrega el mismo valor de entrada. Para entradas mayores a +1, la salida es +1.

[166] Hopfield, J. J. *Neural networks and physical systems with emergent collective computational abilities*, Proceedings of the National Academy of Sciences of the USA, vol. 79 no. 8 pp. 2554-2558, April 1982

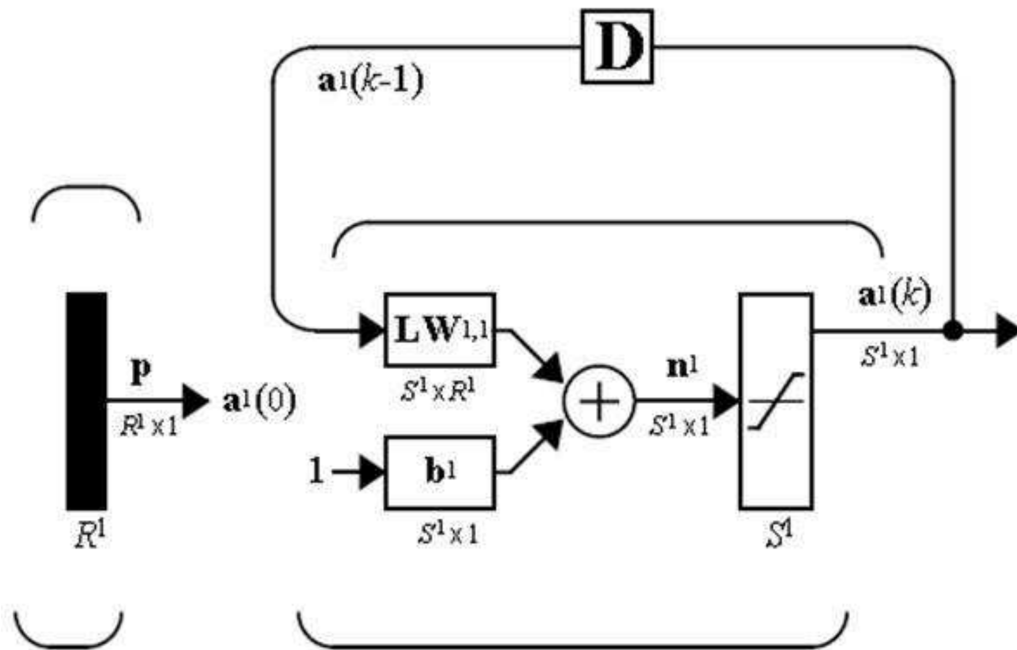


Figura 72. Modelo de Memoria Asociativa de Hopfield

La entrada ***p*** se presenta uno o más vectores ***R-dimensionales*** que establecen las condiciones iniciales para la red. Después de esto, la red produce salidas que son retroalimentadas a las entradas. Este proceso sigue repetidamente, hasta que las salidas de la red se estabilizan. Eventualmente, cada vector de salida converge al vector objetivo aprendido que corresponde al vector de entrada que provocó las iteraciones.

En esencia, las redes Hopfield son importantes sólo desde el punto de vista teórico. Desafortunadamente, las redes Hopfield pueden tener estados estables espurios que generan respuestas erradas.

El *Neural Network Toolbox de MATLAB*, ofrece aplicaciones demostrativas que ayudan a comprender la operación de las redes Hopfield. En el entorno de trabajo de **MATLAB** activar **Help** y en su tabla de contenido seleccionar **Neural Networks Toolbox ► Demos ► Other Demos ► Other Neural Network Design Textbook Demos** ejecutar **Run this demo** para que aparezca la ventana **Neural Network Design** y a través de la tabla de contenido acceder al Capítulo 18, **Hopfield Network**. El código fuente se encuentra en el archivo: **nnd18hn**.

En la siguiente figura, se presenta una vista de la interfaz del ejemplo de aplicación donde se puede interactuar seleccionando puntos en el espacio de entrada (punto azul), para observar la trayectoria de convergencia al estado estable más próximo. La aplicación permite al usuario experimentar con diferentes parámetros de entrenamiento.

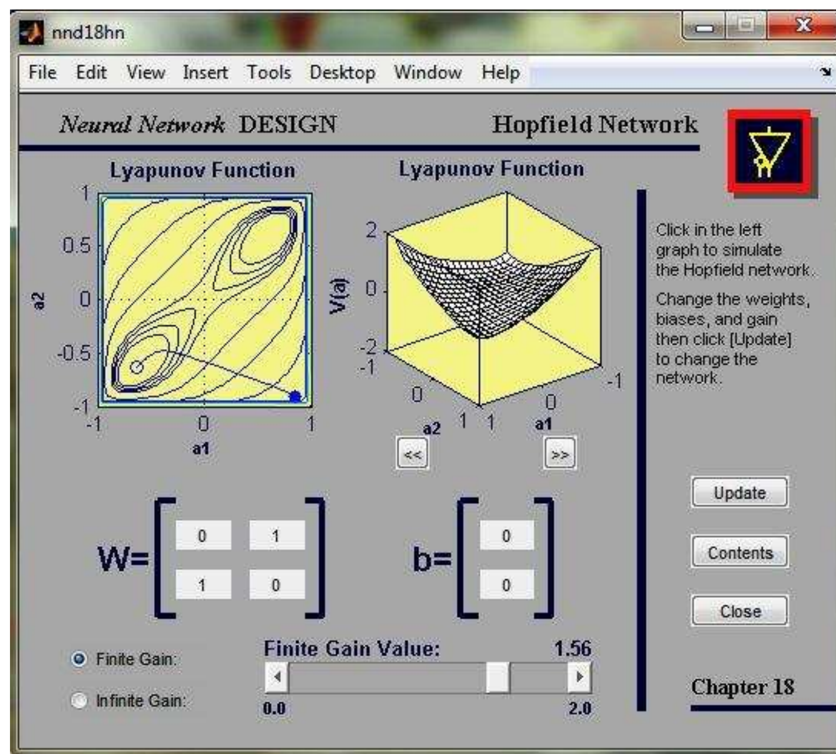


Figura 73. Interfaz de interacción para el ejemplo de Red de Hopfield

9.9.2 Memoria Asociativa Bidireccional

La memoria asociativa bidireccional (BAM) es un tipo de red neuronal recurrente heteroasociativa, propuesta por **Bart Kosko** en 1988[167]. Tiene dos capas: una hace de entrada y la otra de salida. La información fluye entre ellas de forma bidireccional.

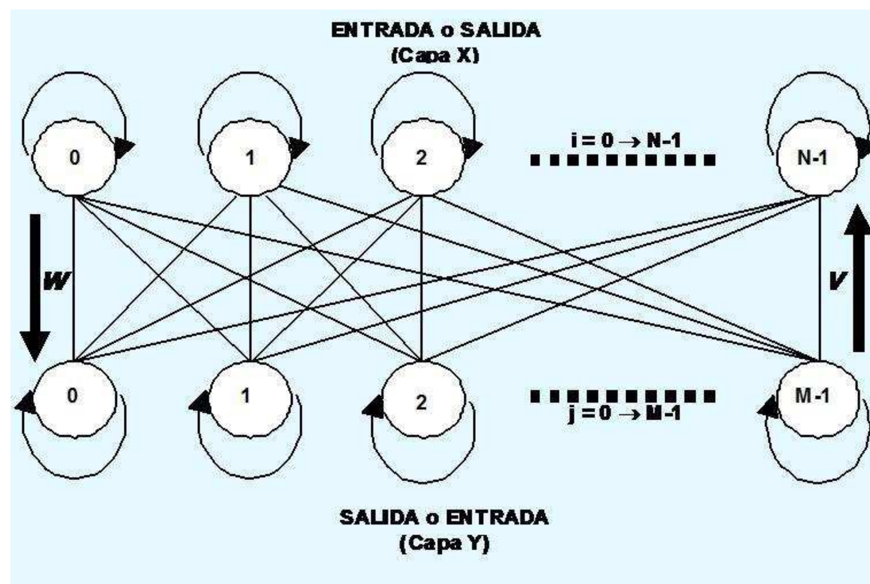


Figura 74. Modelo de Memoria asociativa bidireccional.

---

[167] Kosko, B. *Bidirectional Associative Memories*. IEEE Transactions on Systems, Man and Cybernetics. Vol 18, N° 1, Jan/Feb 1988, pp. 49-60.

Durante la fase de entrenamiento se trata de encontrar la matriz de correlación entre la entrada $\boldsymbol{X}$ y la salida $\boldsymbol{Y}$ (Matriz de Pesos $\boldsymbol{W}$).

$$Y = sign(X * W)$$

Si se tienen $\boldsymbol{P}$ pares entrada salida para entrenar la red, la capa de entrada tiene $\boldsymbol{N}$ nodos y la de salida $\boldsymbol{M}$ nodos, entonces $\boldsymbol{X}$ será una matriz de dimensión $\boldsymbol{P \times N}$, $\boldsymbol{Y}$ una matriz de dimensión $\boldsymbol{P \times M}$ y la matriz de pesos $\boldsymbol{W}$ será de dimensión $\boldsymbol{N \times M}$.

Dado que ***sign(1)=1*** y ***sign(-1)=-1***, se puede decir que ***Y=sign(Y)***. Además, sabiendo que $\boldsymbol{X*X^T}$ es la matriz unitaria, se puede reemplazar, obteniendo: $\boldsymbol{Y=sign(X*X^T*Y)}$, de donde se deduce:

$$W = X^T * Y$$

De igual manera se puede deducir la matriz de pesos $\boldsymbol{V}$, cuando $\boldsymbol{Y}$ es la entrada y $\boldsymbol{X}$ la salida:

$$V = Y^T * X$$

$$X = sign(Y * V)$$

Una aplicación demostrativa, puede encontrarse on-line, así como su código fuente escrito en Perl[168].

9.9.3 Memoria Asociativa Difusa

Bart Kosko al inicio de los 1990, propuso las memorias asociativas difusas. Son arquitecturas de una capa, con alimentación hacia adelante, descritas en términos de una matriz no lineal de productos vectoriales denominados composición $max{-}min$ y una matriz de pesos sinápticos dada por un aprendizaje Hebbiano difuso[169,170].

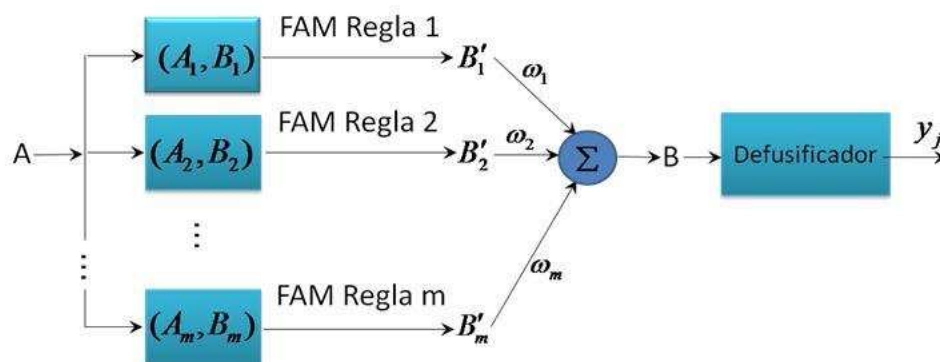


Figura 75. Arquitectura de un Sistema de Memoria Asociativa Difusa (FAM).

---

[168] http://www.lwebzem.com/cgi-bin/res/nnbam.cgi

[169] Kosko, B. *Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence*. Prentice Hall, Englewood Cliffs, N.J., 1992.

[170] Kong, S-G and Kosko, B. *Adaptive fuzzy systems for backing up a truck-and-trailer*. IEEE Transactions on Neural Networks Vol. 3, N° 2, Mar. 1992, pp. 211–223.

Una de las principales desventajas del modelo propuesto por Kosko es la gran cantidad de interferencia cruzada que da lugar a que posea una muy baja capacidad de almacenamiento. En 1996, **Fu-lai Chung** y **Tong Lee**, proponen una nueva arquitectura con capacidad para almacenar múltiples reglas[171]; y, en el 2006, **Peter Sussner** y **Marcos Eduardo Valle**, presentan una nueva clase de memoria asociativa neuronal, basada en la teoría de conjuntos difusos la ***Implicative Fuzzy Associative Memory*** (***IFAM***)[172].

Una ***IFAM*** consiste de una red completamente interconectada de neuronas lógicas **Pedrycz**[173], con umbrales de decisión, cuyos pesos son determinados por el mínimo de implicaciones de activaciones pre sinápticas y pos sinápticas. Entre los resultados presentados por los autores, para los modelos autoasociativos, se incluyen: convergencia en una pasada, almacenamiento ilimitado y tolerancia a patrones incompletos.

9.9.4 Redes Adaptativas de Neuronas Lineales

En 1962, **Ted Hoff** como parte de disertación para obtener el grado de PhD en Stanford University y el profesor **Bernard Widrow**, coinventaron el filtro de mínimos cuadrados, basado en un elemento adaptativo lineal (***Ada****ptive* ***Lin****ear* ***E****lement*, **ADALINE**). La principal referencia en este campo es un libro publicado por **Widrow** y **Sterns**[174], en 1985.

9.9.4.1 ADALINE

El ADALINE es similar al perceptrón, con la diferencia que la función de transferencia en lugar de ser binaria, es lineal.

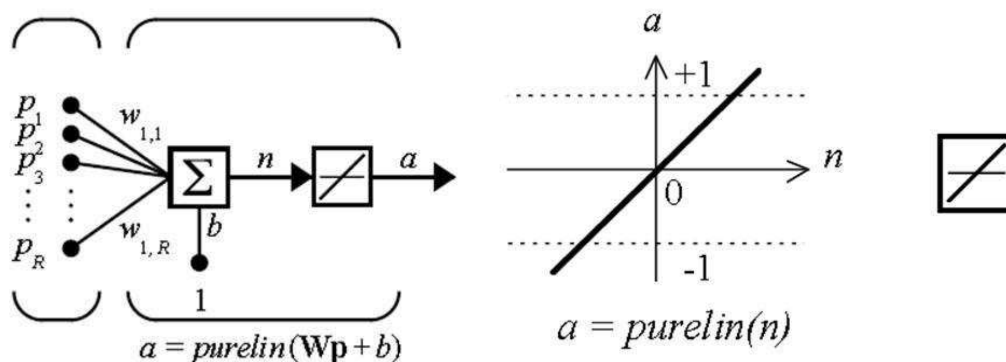


Figura 76. Modelo del ADALINE y Función de transferencia, usados por MATLAB

Si bien sólo pueden resolver problemas linealmente separables, su regla de aprendizaje basada en la minimización de la media de los mínimos cuadrados (***L****east* ***M****ean* ***S****quares*) del error, es mucho más poderosa que la regla de aprendizaje del perceptrón.

---

[171] Chung, F, and Lee, T. On fuzzy associative memory with multiple-rule storage capacity. IEEE Transactions on Fuzzy Systems Vol. 4, N° 3, Aug. 1996, pp. 375–384.
[172] Sussner, P. and Valle, M. E. *Implicative Fuzzy Associative Memories*. IEEE TRANSACTIONS ON FUZZY SYSTEMS, Vol. 14, N°. 6, Dec. 2006.
[173] Pedrycz, W and Hirota, K. *Uninorm-Based Logic Neurons as Adaptive and Interpretable Processing Constructs*. Soft Comput., 2007: pp. 41-52
[174] Widrow B, and S. D. Sterns. *Adaptive Signal Processing*. Prentice Hall, NY, 1985.

9.9.4.2 Algoritmo de Aprendizaje de Widrow – Hoff

El algoritmo ***LMS*** está basado en una aproximación del procedimiento de descenso pronunciado. Estimando el error medio cuadrático en cada iteración, es posible realizar un proceso de optimización, basado en el cálculo diferencial:

$$\text{Dado un conjunto de Q pares entrada - salida :}$$
$$\{p_1,t_1\},\{p_2,t_2\},\ldots,\{p_Q,t_Q\}$$
$$\text{Se puede definir el error medio cuadrático (mse) como :}$$
$$mse = \frac{1}{Q}\sum_{k=1}^{k=Q} e(k)^2 = \frac{1}{Q}\sum_{k=1}^{k=Q}(t(k)-a(k))^2$$

Tomando la derivada parcial del error elevado al cuadrado, con respecto a los pesos y al umbral de decisión, se tiene:

$$\frac{\partial e(k)^2}{\partial w_{1,j}} = 2e(k)\frac{\partial e(k)}{\partial w_{1,j}} \quad \text{Para } j = 1,2,\ldots,R$$
$$\frac{\partial e(k)^2}{\partial b} = 2e(k)\frac{\partial e(k)}{\partial b}$$

Ahora, expandiendo las derivadas parciales del error:

$$\frac{\partial e(k)}{\partial w_{1,j}} = \frac{\partial [t(k)-a(k)]}{\partial w_{1,j}} = \frac{\partial}{\partial w_{1,j}}\left[t(k) - \sum_{i=1}^{R} w_{1,i}\, p_i(k) + b\right]$$
$$\text{De donde, se tiene que :}$$
$$\frac{\partial e(k)}{\partial w_{1,j}} = -p_j(k);\ y,$$
$$\frac{\partial e(k)}{\partial b} = -1$$

Finalmente, se puede determinar los cambios en los pesos y en el valor del umbral de decisión de la neurona.

$$\delta w_{i,j} = \alpha \frac{\partial e^2(k)}{\partial w_{1,j}} = -2\alpha . e(k). p_j(k)$$
$$\delta b = -\alpha . e(k)$$
$$\text{Donde } \alpha \text{ es un factor de aprendizaj e } 0 < \alpha \le 1$$

De aquí se puede generalizar y obtener las matrices de ajuste de pesos y el vector de ajuste de los umbrales de decisión, para el caso en que se tengan múltiples neuronas:

$$\mathbf{W}(k+1) = \mathbf{W}(k) - 2\alpha.\mathbf{e}(k).\mathbf{p}^T(k)$$
$$\mathbf{b}(k+1) = \mathbf{b}(k) - 2\alpha.\mathbf{e}(k)$$

El algoritmo ***LMS*** dio lugar al desarrollo de aplicaciones del ADALINE, de redes adaptativas con neuronas lineales (**MADALINE**) y al algoritmo de retroprogación del error utilizado en redes neuronales con arquitectura multicapa (*backpropagation networks*).

La neurona lineal, puede ser entrenada para realizar clasificaciones lineales, actuar como filtro adaptativo, para aprender una función afín a sus entradas o para encontrar una aproximación a una función no lineal.

9.9.4.3 Filtro Adaptativo

El filtraje adaptativo es una de las aplicaciones prácticas más importantes de los ADALINE. Para esto se requiere de una línea de retardo con tomas secuenciales (***tapped delay line, TDL***). En el modelo dado en la siguiente Figura, la señal de entrada ingresa por la parte superior izquierda y pasa a través de los **`N-1`** retardos. La salida de las tomas de la **TDL** es un vector **`N-dimensional`**, conformado por la señal de entrada actual, la señal anterior y así sucesivamente. Al combinar la **TDL** con una ADALINE, se crea un filtro adaptativo tipo **FIR** (***Finite Implulse Response***)[175].

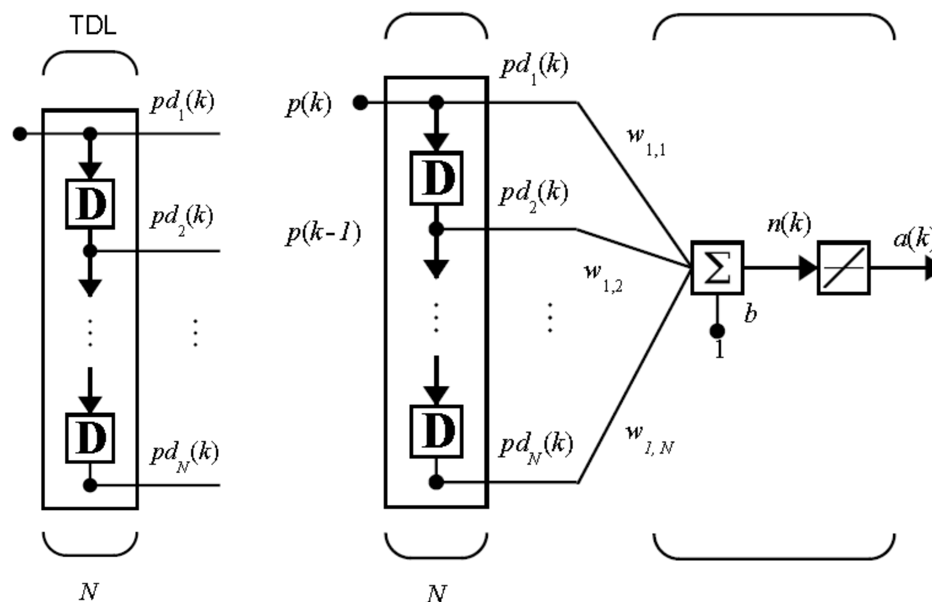


Figura 77. Modelo de TDL y Filtro Adaptativo tipo FIR

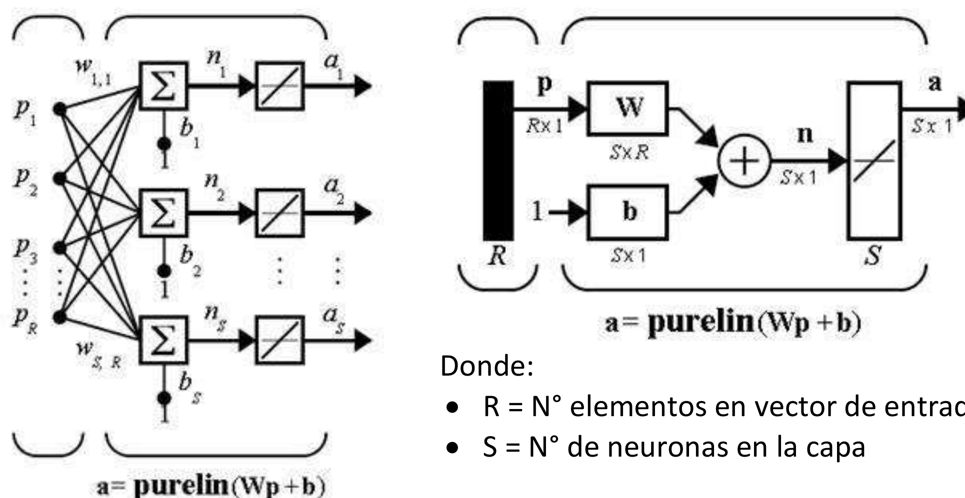
Para experimentar con varias aplicaciones de los ADALINE, se recomienda utilizar los ejemplos proporcionados por MATLAB: **Neural Network Toolbox ► Demos ► Linear Networks**.

9.9.4.4 Red MADALINE

Si se conectan varias neuronas lineales, formando una capa, se obtiene una arquitectura denominada **MADALINE** (*Many* **ADALINE**).

---

[175] http://idlastro.gsfc.nasa.gov/idl_html_help/Finite_Impulse_Response_(FIR)_Filters.html



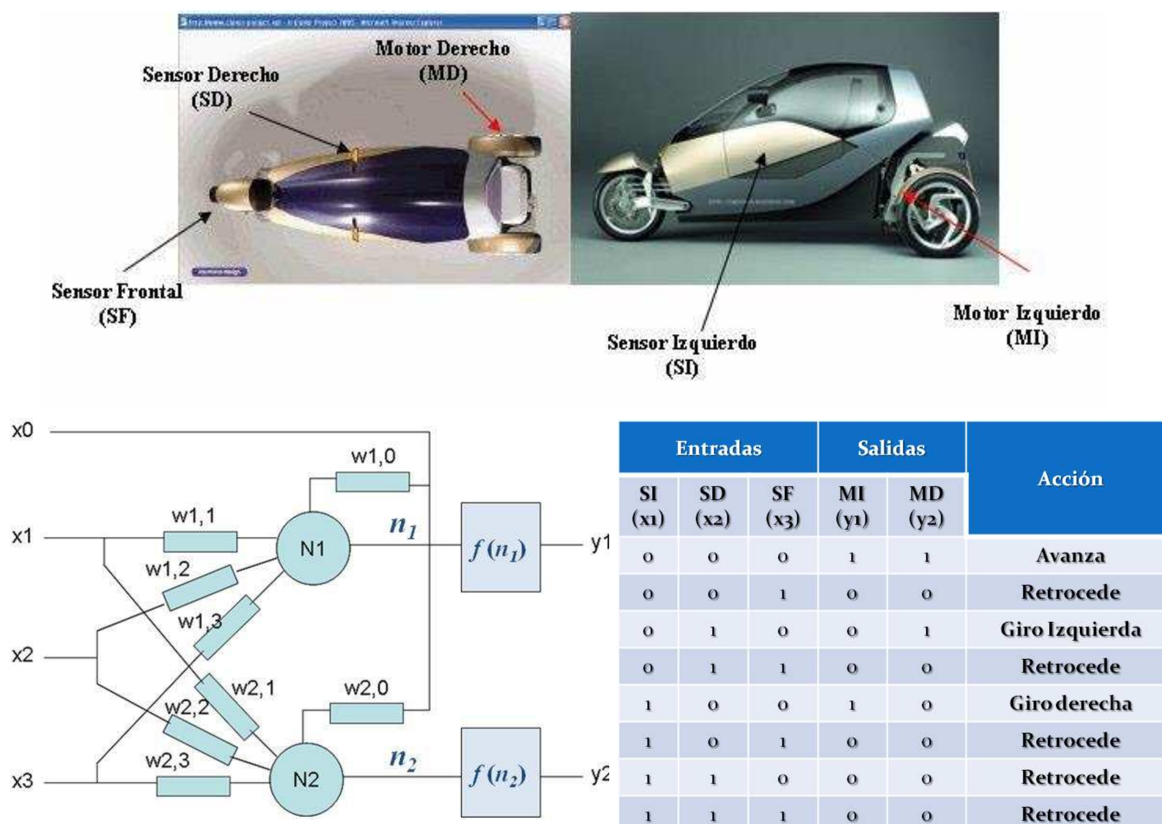
Donde:

- R = N° elementos en vector de entrada
- S = N° de neuronas en la capa

Figura 78. Arquitectura de Red MADALINE y Representación usados por MATLAB

También se pueden realizar otras aplicaciones, incluyendo filtros adaptativos más complejos, con arquitecturas MADALINE.

El siguiente es un ejemplo de una red MADALINE para el control de un vehículo robótico capaz de mantenerse en movimiento evadiendo obstáculos en su camino. Tiene una rueda delantera y un motor en cada rueda trasera, para controlar mediante tracción diferencial, el avance, giro o retroceso del vehículo.



Entradas			Salidas		Acción
SI (x1)	SD (x2)	SF (x3)	MI (y1)	MD (y2)	
0	0	0	1	1	Avanza
0	0	1	0	0	Retrocede
0	1	0	0	1	Giro Izquierda
0	1	1	0	0	Retrocede
1	0	0	1	0	Giro derecha
1	0	1	0	0	Retrocede
1	1	0	0	0	Retrocede
1	1	1	0	0	Retrocede

Figura 79. Vehículo Robótico, red MADALINE y Tabla de Verdad para acciones de control.

Las señales de los sensores de proximidad son acondicionadas para que al detectar obstáculo se pongan en 1 lógico, caso contrario están 0 lógico. La entrada X0 a los umbrales de decisión es igual a 1 lógico. Las señales de salida de la red, también se acondicionan y codifican para controlar el sentido de giro de los motores de las llantas posteriores. Con una señal de 1 lógico, la rueda gira en el sentido de avanzar al vehículo y con 0 lógico gira en el sentido de retroceder. El torque de giro a la izquierda se genera cuando la rueda izquierda gira hacia atrás y la derecha gira hacia adelante (tracción diferencial). En un micro controlador (p.e. ATmega 16[176]) se puede simular la red y programar las acciones de acondicionamiento de señales y el control de los motores.

9.9.5 Redes de Retropropagación (*Backpropagation*)

En principio una red de retropropagación puede contener un número arbitrario de capas de neuronas, entre la capa de entrada y la de salida, pero sólo simples casos han sido estudiados a profundidad. Para empezar, se considerará el caso de una simple red, en la que la capa de entrada alimenta directamente a la capa de salida, sin intervención de capas de neuronas *internas* u *ocultas*.

Supóngase que se tienen N entradas cuyos estados comprendidos entre ± 1, están representados por la expresión x_k, $(k = 0 \; .. \; N-1)$. El número de salidas se suponen iguales a M, con estados dados por y_i, $(i = 0, \; .... \; , M-1)$. La activación de cada una de las neuronas de salida debido a las señales aplicadas por la capa de entrada, es una función del campo local h_i:

$$y_i = f(h_i) = f(\sum_{k=0}^{N-1} \omega_{ik} \cdot x_k)$$

La función de activación que usualmente se utiliza para las neuronas de salida es estocástica, del tipo:

$$f(h) = \tanh(\beta h) = \frac{1 - e^{-2\beta.h}}{1 + e^{-2\beta.h}}$$

$$\beta = \frac{1}{T}$$

La tarea que resta es escoger las conexiones sinápticas ω_{ik} de tal manera que una cierta entrada x_k produzca la reacción deseada, especificada por los correctos estados de las neuronas de salida (d_i). Por cierto, esta condición no solo tiene que ser satisfecha para una entrada, sino para un cierto número de patrones P, enumerados por el exponente p:

$$y_i = f(h_i^p) = f(\sum_{k=0}^{N-1} \omega_{ik}.x_k^p) \quad p = 0..P-1$$

[176] http://atmel.com/dyn/products/product_card.asp?part_id=2010

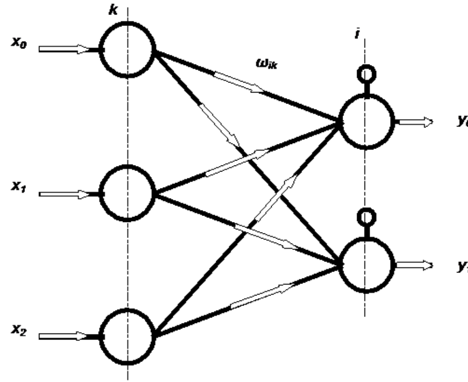


Figura 80. Simple red de retropropagación.

No se conoce una función explícita que permita calcular directamente las conexiones sinápticas ω_{ik} a partir de las entradas x_k^p. Sin embargo, es posible construir procedimientos iterativos que convergen a los valores deseados para las conexiones sinápticas, en caso de que estos existan.

9.9.5.1 Aprendizaje por Gradiente

La Regla de Aprendizaje por Gradiente *o* Regla Delta fue desarrollada para redes de retropropagación del error de dos capas, y constituye la base para generalizar para redes más complejas con capas escondidas. Para este propósito, se inicia el análisis con la expresión que define la desviación absoluta total entre la salida actual y la correcta, y se aplica un método de minimización, basado en cálculo diferencial.

$$D = \frac{1}{2} \cdot \sum_{p=0}^{P-1} \sum_{i=0}^{M-1} (d_i^p - y_i^p)^2$$

Introduciendo la función de activación, se tiene:

$$D = \frac{1}{2} \cdot \sum_p \sum_i [\, d_i^p - f(h_i^p)\,]^2 = \frac{1}{2} \cdot \sum_p \sum_i [\, d_i^p - f(\sum_k \omega_{ik} \cdot x_k^p)\,]^2$$

Ahora se calcula el gradiente de *D* con respecto a los acoplamientos sinápticos:

$$\frac{\partial D}{\partial \omega_{ik}} = -\sum_p [\, d_i^p - f(h_i^p)\,] . f'(h_i^p) . \frac{\partial h_i^p}{\partial \omega_{ik}} = -\sum_p \Delta_i^p \cdot x_k^p$$

Donde,

$$\Delta_i^p = [\, d_i^p - f(h_i^p)\,] . f'(h_i^p) = [\, d_i^p - y_i^p\,] . f'(h_i^p)$$

La desviación *D* decrece si las conexiones sinápticas se varían utilizando la expresión:

$$\Delta \omega_{ik} = -\alpha \frac{\partial D}{\partial \omega_{ik}} = \alpha \sum_p \Delta_i^p \cdot x_k^p$$

El factor de aprendizaje η debe ser escogido lo suficientemente pequeño como para conseguir una convergencia muy cercana al valor deseado para las conexiones sinápticas.

La regla de Aprendizaje por Gradiente no se puede aplicar a redes constituidas por neuronas determinísticas con funciones de activación discontinuas debido a que se requiere la derivada de la función de transferencia, pero se puede generalizar muy fácilmente para redes con más de dos capas.

9.9.5.2 Aprendizaje por Retropropagación del Error

Para utilizar redes multicapa en forma efectiva, es necesario disponer de un método que determine los acoplamientos sinápticos y los umbrales de decisión.

En 1969, en una publicación de **Bryson y Ho**[177], se menciona por primera vez los fundamentos del algoritmo por **retroprogación del error** (***error backpropagation***), sin embargo no fue sino hasta 1974 con la disertación de **Paul Werbos**[178] y posteriormente con el trabajo desarrollado y publicado por otros investigadores[179] que se logró imponer este método para el entrenamiento de redes neuronales multicapa. Como se indicó anteriormente, está basado en la generalización de la Regla Delta.

Para el análisis, sin perder generalidad, supóngase que se tiene una red de tres capas tal como la que se esquematiza en la siguiente figura.

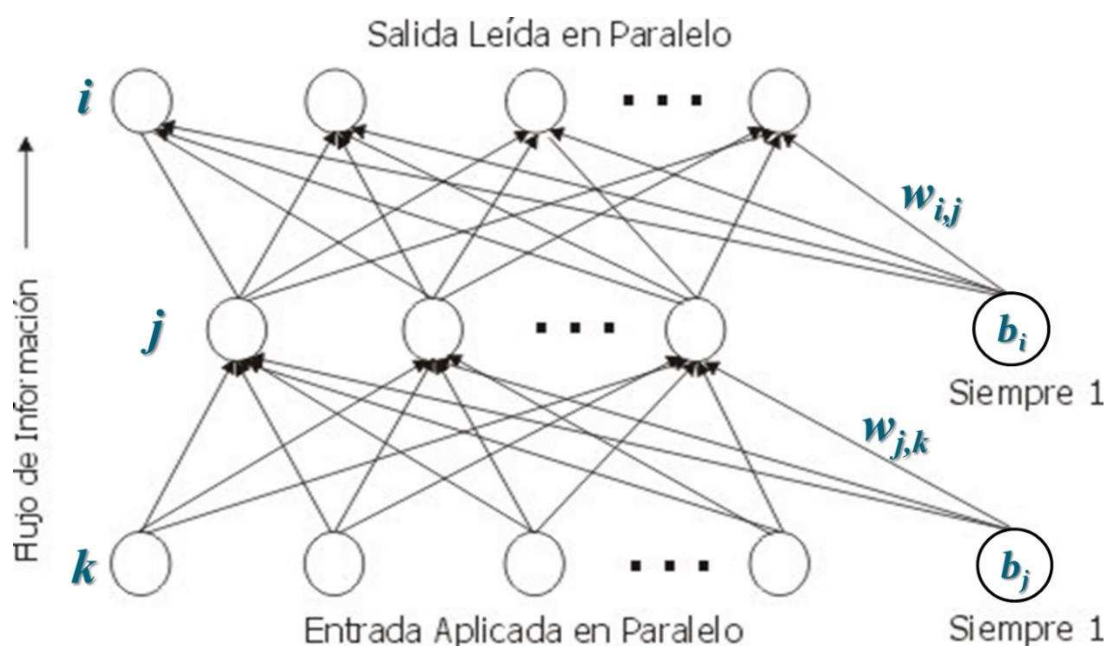


Figura 81. Esquema de una red *backpropation* de tres capas

---

[177] Bryson, A.E.; Ho, Y.C. *Applied optimal control*. Blaisdel, NW, USA, 1969.

[178] Werbos, P. J. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD thesis, Harvard University, 1974

[179] Rumelhart, David E.; Hinton, Geoffrey E., Williams, Ronald J. *Learning representations by back-propagating errors*. Nature 323 (6088), 8 Oct 1986: pp. 533–536.

Las ecuaciones que gobiernan los estados de una red de tres capas, son las siguientes:

$$y_i^p = f\,(h_i^p)$$

$$h_i^p = \sum_j \omega_{ij} \cdot x_j^p - \theta_i$$

$$x_j^p = f\,(\bar{h}_j^p)$$

$$\bar{h}_j^p = \sum_k \bar{\omega}_{jk} \cdot x_k^p - \bar{\theta}_j$$

Las conexiones sinápticas y los umbrales de comparación, deben ser ajustados de tal forma que la desviación absoluta entre la salida actual y la deseada se haga tan pequeña como sea posible.

$$D(\omega_{ij}, \theta_i, \bar{\omega}_{jk}, \bar{\theta}_j) = \frac{1}{2} \sum_p \sum_i [\, d_i^p - f(h_i^p)\,]^2$$

Básicamente lo que se trata es de buscar el mínimo (idealmente debería ser el mínimo global) de la superficie de error definida por la ecuación anterior.

Para este propósito se computa el gradiente de *D* con respecto a cada uno de sus variables y se determinan las prescripciones para ajustar sus valores en forma iterativa. Para el caso de conexiones entre la capa de salida y la capa escondida, se tienen las siguientes relaciones:

$$\Delta \omega_{ij} = -\alpha \frac{\partial D}{\partial \omega_{ij}} = \alpha \sum_p [\, d_i^p - f(h_i^p)] \,.\, f'(h_i^p) . \frac{\partial h_i^p}{\partial \omega_{ij}} = \alpha \sum_p \Delta_i^p \cdot x_j^p$$

$$\Delta \theta_i = -\alpha \frac{\partial D}{\partial \theta_i} = \alpha \sum_p [\, d_i^p - f(h_i^p)] \,.\, f'(h_i^p) . \frac{\partial h_i^p}{\partial \theta_i} = -\alpha \sum_p \Delta_i^p$$

Donde,

$$\Delta_i^p = [\, d_i^p - f(h_i^p)] \,.\, f'(h_i^p)$$

Aplicando un procedimiento similar, se obtienen las prescripciones para el ajuste de los parámetros asociados con las conexiones sinápticas entre la capa escondida y la capa de entrada:

$$\Delta \bar{\omega}_{jk} = -\alpha \frac{\partial D}{\partial \bar{\omega}_{jk}} = \alpha \sum_{p,i} [\, d_i^p - f(h_i^p)] \,.\, f'(h_i^p) . \frac{\partial h_i^p}{\partial x_j^p} . \frac{\partial x_j}{\partial \bar{\omega}_{jk}} =$$

$$= \alpha \sum_{p,i} \Delta_i^p \cdot \omega_{ij} \,.\, f'(\bar{h}_j^p) . \frac{\partial \bar{h}_j}{\partial \bar{\omega}_{jk}} = \alpha \sum_p \bar{\Delta}_j^p \cdot x_k^p$$

$$\Delta\theta_j = -\alpha \frac{\partial D}{\partial \bar{\theta}_j} = \alpha \sum_{p,i} [\, d_i^p - f(h_i^p)] \,.\, f'(h_i^p) \,.\, \frac{\partial h_i^p}{\partial x_j^p} \,.\, \frac{\partial x_j}{\partial \bar{\theta}_j} =$$

$$= \alpha \sum_{p,i} \Delta_i^p \, \omega_{ij} \,.\, f'(\bar{h}_j^p) \,.\, \frac{\partial \bar{h}_j}{\partial \bar{\theta}_j} = -\alpha \sum_{p} \bar{\Delta}_j^p$$

Donde,

$$\bar{\Delta}_j^p = (\sum_i \Delta_i^p \,.\, \omega_{ij}) \,.\, f'(\bar{h}_j^p)$$

La ecuación última, indica que el esquema de **Corrección del Error** trabaja propagando la información que especifica el grado de desviación que tiene la salida de la red, con respecto a la salida deseada, hacia atrás a través de la red, en contra de la dirección de las conexiones sinápticas.

Como se ha podido demostrar, el algoritmo para entrenar redes con capas ocultas es relativamente fácil de implantar. Sin embargo, no existe ningún teorema que garantice la convergencia en este tipo de redes.

El proceso de entrenamiento consiste en una secuencia de *épocas* en las cuales el conjunto de patrones x_k^p a ser aprendidos es presentado a la red. Cuando las correcciones a las conexiones sinápticas y a los valores de umbral de decisión son acumuladas y aplicadas al final de una época, se dice que el algoritmo es tipo *batch learning*. Si los parámetros de la red son actualizados después de cada presentación de un patrón, el algoritmo se denomina *incremental learning*. Esta forma de entrenamiento en ciertos casos puede mejorar la rapidez de aprendizaje.

Una importante modificación que tiene el potencial de mejorar la estabilidad del proceso de entrenamiento consiste en introducir una especie de efecto de histéresis o *momentum*. En ciertas regiones en las que la superficie que representa el error tiene fuertes curvaturas los términos Δ del gradiente pueden hacerse muy grandes.

A no ser que el factor de aprendizaje α se lo haga muy pequeño, las correcciones sinápticas tenderán a hacer oscilar el ajuste de parámetros ocasionando demora, o peor aun falla en la convergencia del algoritmo de entrenamiento. Para evitar este problema, se agrega al término de corrección una memoria tal que no esté sujeto a cambios abruptos, por ejemplo:

$$\Delta\omega_{ij}^{(n)} = \alpha \sum_p \Delta_i^p \,.\, x_j^p + \beta \,.\, \Delta\omega_{ij}^{(n-1)}$$

Donde *n* es el número de la época de entrenamiento y β es el *momentum*. Desafortunadamente no existe un criterio general para indicar en qué forma deben escogerse los parámetros α y β. Los valores óptimos dependerán de cada problema.

9.9.5.3 Clasificación de Cangrejos

En MATLAB se encuentra el siguiente ejemplo. Se trata de clasificar cangrejos por su género (masculino, femenino) a partir de características físicas. Se utilizan seis mediciones que corresponden a los siguientes atributos:



Figura 82. Cangrejo de mar

- Especie (0 = azul; 1 = anaranjado)
- Tamaño del lóbulo frontal
- Ancho posterior
- Longitud del carapacho
- Ancho del carapacho
- Profundidad del cuerpo
- Género (male, female)

9.9.5.3.1 Preparación de los Datos

Dada la naturaleza no lineal de los problemas que se encuentran en el mundo real, las redes neuronales son las mejores herramientas que uno puede utilizar para resolver problemas como este que trata de clasificar cangrejos por su género, utilizando las seis mediciones de características como entradas y el género como salida.

Los datos recopilados corresponden a 200 ejemplares (`crabdata.csv`[180]). Las mediciones de los atributos, tienen valores numéricos, pero el género está dado en forma textual. Debido a que las redes neuronales en MATLAB no pueden ser entrenadas con datos no numéricos, es necesario aplicar una técnica de conversión simbólica a codificación unaria. Esto es, 'Female' será representado por el vector [1 0] y 'Male' representado por [0 1].

9.9.5.3.2 Justificación del tipo de red

Para la clasificación se utiliza una red neuronal tipo *feedforward backpropagation* con una configuración 6:20:2 (`crabclassify.m`[181]).

Los vectores de entrada que contienen los 6 atributos, son transpuestos antes de ser aplicados a la red, igual ocurre con los vectores de salida que corresponden al género de los cangrejos. A manera de ejercicio, se recomienda:

- Analizar el problema y el archivo de datos correspondiente.
- Estudiar el script de MATLAB, comprender y describir la operación de las diferentes funciones utilizadas.
- Interpretar los resultados obtenidos.
- Obtener conclusiones y recomendaciones.

---

180 C: \Archivos de programa\MATLAB\R2010b\toolbox\nnet\nndemos\private\crabdata.csv

181 C: \Archivos de programa\MATLAB\R2010b\toolbox\nnet\nndemos\crabclassify.m

9.9.5.3.3 Entrenamiento y Prueba de la red:

Debido a que la red neuronal inicializa sus pesos con valores aleatorios, cada vez que se pruebe la red neuronal puede dar resultados diferentes. Para evitar esto, se utiliza un valor semilla para evitar la aleatoriedad. Sin embargo esto no es necesario para todas las aplicaciones.

El algoritmo automáticamente elige un conjunto para el entrenamiento y deja fuera un conjunto aleatorio para las pruebas. Esto se registra en una estructura de datos, en el campo (**tr.testInd**).

Finalmente, se obtienen los resultados de simulación y se construye la matriz de confusión para determinar el desempeño del clasificador.

9.9.5.4 Predicción de Consumo de Energía Eléctrica

Este es un ejercicio adaptado de un proyecto de titulación presentado en la Facultad de Ingeniería Eléctrica de la Universidad Tecnológica de Pereira, Colombia[182].

Un sistema de energía eléctrica debe abastecer de energía a todos los puntos de carga con una buena calidad del servicio. Por lo tanto un sistema eléctrico confiable, el cual asegura buena calidad, debe contar con las siguientes características:

- Entregar energía en forma continua a todos los puntos de carga.
- Los límites de la frecuencia y la tensión deben estar dentro de valores tolerables.
- El sistema debe operar en la medida de lo posible, con costos mínimos y con un mínimo de alteraciones ambientales o ecológicas.

Estas características pueden adquirirse por medio de una planeación exhaustiva del sistema, que permita conocer no solo su estado actual, sino también las medidas que deben adoptarse para condiciones futuras.

Una de las herramientas útiles en el planeamiento de un sistema eléctrico es la predicción del consumo de carga, la cual permite conocer de antemano la necesidad de expansión del sistema; la finalidad de la predicción siempre será el mejoramiento del servicio, convirtiéndose en uno de los primeros pasos en cualquier proceso de planeamiento de un sistema eléctrico.

Al hablar de predicción de carga, resulta útil aclarar que como carga se asume todo equipo que demanda alimentación del sistema de energía eléctrica, tales como lámparas, electrodomésticos, motores eléctricos, hornos eléctricos, etc. De esta manera se tienen varios tipos de carga:

---

[182] Acosta, M. I. y Zuluaga, C. A. *Tutorial sobre Redes Neuronales Aplicadas en Ingeniería Eléctrica y su Implementación en un sitio WEB*. Proyecto de grado para título de Ingeniero Electricista. Facultad de Ingeniería Eléctrica. Universidad Tecnológica de Pereira, Colombia, Oct. 2000. **http://ohm.utp.edu.co/neuronales**

- Motores en general
- Equipos de calentamiento
- Equipos electrónicos
- Equipo de iluminación

Desde el punto de vista del sistema de energía eléctrica, las cargas pueden ser separadas en tres grupos funcionales:

- Cargas domiciliarias
- Cargas industriales
- Cargas comerciales

Estas cargas presentan características muy diferentes con relación al tamaño, simetría (monofásica o trifásica), constancia de la carga y el período de funcionamiento.

La predicción de consumo de carga refleja las necesidades futuras de una población; esta previsión debe ser lo más ajustada a la realidad, ya que unos valores inferiores a los reales causaran deficiencias en la prestación del servicio en el futuro y un pronóstico de necesidades superior al real, motiva la inversión prematura en instalaciones que no tendrán un aprovechamiento inmediato.

La proyección del suministro de energía se hace con base en el consumo, aplicando porcentajes de pérdidas que pueden obtenerse de un análisis de los registros históricos (que normalmente se presentan en forma estadística), o por similitud con otros sistemas. En general las pérdidas tienden a disminuir a causa de las mejoras progresivas, que se introducen en los sistemas de transmisión, subtransmisión y distribución. En forma similar al consumo de energía, la proyección de la demanda pico se obtiene sumando las demandas máximas coincidentes en hora pico.

En la siguiente Figura se ha tabulado el comportamiento típico de consumo de carga de una población, a la cual se realizó un seguimiento hora a hora durante una semana. Se ha tomado el Lunes como 1 y en su orden el día Domingo como 7, se asumió la una de la mañana como la hora 1 y las doce de la noche como la hora 24. Los datos correspondientes al consumo máximo de energía eléctrica en la hora determinada, se encuentran en (MW).

HORA	Lunes	Martes	Miércoles	Jueves	Viernes	Sábado	Domingo
1	1,23	1,6049	1,663	1,7299	1,7129	1,713	1,414
2	1,0889	1,4389	1,4689	1,5159	1,4569	1,2821	1,325
3	1,0289	1,3631	1,389	1,432	1,3461	1,182	1,2249
4	0,9879	1,3559	1,3751	1,3931	1,288	1,122	1,2239
5	0,9879	1,3439	1,3611	1,3909	1,2331	1,0961	1,124
6	1,105	1,389	1,414	1,431	1,1911	1,1059	1,0191
7	1,3729	1,5699	1,604	1,614	1,157	1,171	0,9989
8	1,6649	1,775	1,8009	1,817	1,17	1,2751	0,9989
9	1,79	2,018	2,0739	2,0989	1,2139	1,4121	0,979
10	2,1569	2,19	2,2301	2,226	1,337	1,545	1,015
11	2,323	2,3359	2,3649	2,381	1,4799	1,711	1,1271
12	2,3659	2,363	2,399	2,3741	1,574	1,741	1,2271
13	2,3731	2,3359	2,358	2,3021	1,5951	1,7129	1,296
14	2,2311	2,156	2,2	2,1459	1,5771	1,62	1,313
15	2,156	2,0799	2,1231	2,0581	1,5629	1,557	1,2909
16	2,208	2,1651	2,1749	2,0809	1,532	1,5831	1,26
17	2,2949	2,2551	2,2049	2,1651	1,544	1,6251	1,2669
18	2,3741	2,3671	2,3349	2,238	1,638	1,6251	1,3631
19	2,5	2,477	2,464	2,282	1,731	1,895	1,453
20	2,434	2,431	2,378	2,154	1,748	1,904	1,602
21	2,356	2,354	2,414	2,102	1,7921	1,931	1,644
22	2	2,21	2,004	1,995	1,8321	1,936	1,615
23	1,989	1,7085	1,8582	1,904	1,862	1,958	1,503
24	1,808	1,7	1,7071	1,859	1,793	1,748	1,499

Figura 83. Tabla de Datos de consumo de carga de una población durante una semana

Estos datos son una recopilación de un promedio histórico de diferentes años, del consumo de la población escogida como muestra, con ellos se puede determinar la tendencia del consumo de los usuarios de esta población. El objetivo es entrenar una red neuronal que aprenda los datos anteriores y a partir de ellos esté en capacidad de predecir lo que sucederá en cualquier día de la semana a una hora determinada en años futuros.

El comportamiento de estos usuarios se visualiza en la siguiente Figura en donde puede notarse las tendencias que pueden llegar a causar congestión en la central en caso de no preverse con anticipación:

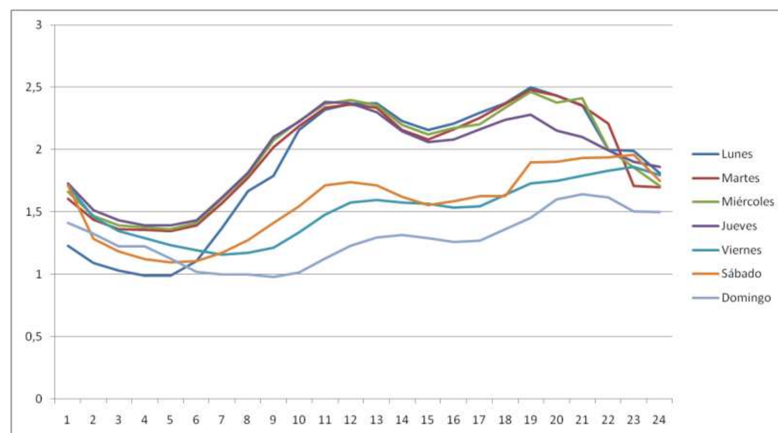


Figura 84. Curvas de Carga

9.9.5.4.1 Tipo de Red

El algoritmo que entrena la red *backpropagation* es de aprendizaje supervisado, el cual necesita conocer cuál es la salida esperada (columnas interiores de la Tabla de Datos) asociada a cada una de las entradas (columna referente al día y fila referente a la hora en la Tabla de Datos), que le sirven para actualizar pesos y umbrales de decisión.

Una de las mayores ventajas de las redes multicapa, y en especial de la red *backpropagation*, es que pueden aproximar cualquier función si se escoge una adecuada configuración para la red y un adecuado número de neuronas en la capa oculta. Esto depende de la experiencia del desarrollador de la red ya que es imposible determinar una configuración exacta de la red para cada aplicación.

El proceso de aprendizaje no es fijo para ninguna red neuronal, el éxito consiste en probar con diferentes configuraciones hasta obtener la respuesta deseada. Para esta aplicación se sugiere probar con una red 2:12:8:1, es decir que para un vector de entrada de dos dimensiones (Día de la Semana y Hora del Día), se tiene una sola salida de red (Consumo correspondiente), se tienen 12 neuronas en la primera capa oculta y 8 neuronas en la segunda capa oculta.

Un esquema de la red puede observarse en la siguiente Figura, la cual muestra las características de los datos de entrada y salida de la red. Allí puede verse que es necesario ingresar a la red el día y la hora para los cuales se desea conocer el valor de demanda pico de acuerdo a la convención de la Tabla de datos, las condiciones de entrada a la red pueden ser tan complejas como se quiera, es decir, la entrada puede convertirse fácilmente en un vector de cuatro componentes que incluya además del día y la hora, el mes y el año para los cuales se requiere predecir el valor de demanda; existen también otras opciones como convertir la entrada en un vector de tres componentes donde se incluya hora, día y tipo de carga para el cual se desea prever el consumo.

La elección de los patrones de entrada debe realizarse dependiendo de las necesidades explícitas que se tengan en el momento de hacer la predicción de carga, de la forma en que vaya a procesarse la información de salida de la red, de la cantidad y calidad de la información disponible.

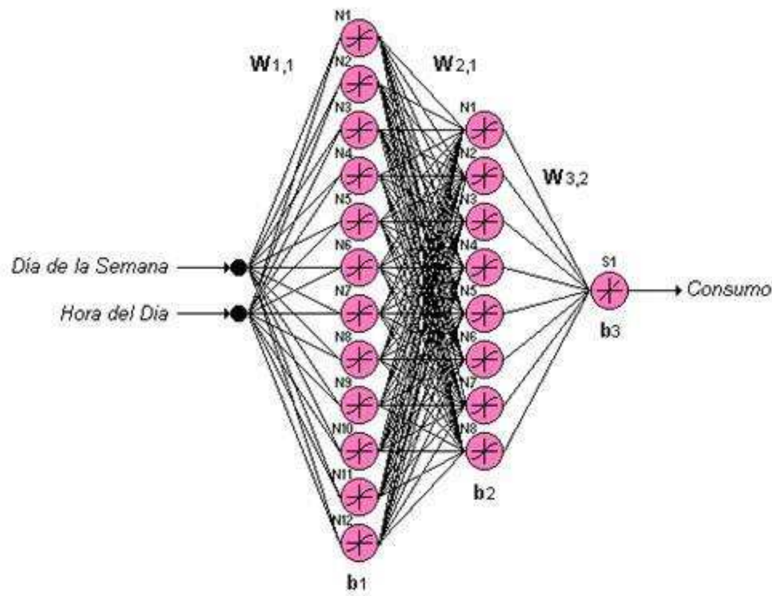


Figura 85. Red Backpropagation 2:12:8:1 para predicción del consumo de energía eléctrica

Cualquier cambio que se realice en los patrones de entrenamiento exige una codificación diferente del vector de entrada y a su vez cambia las condiciones generales de la red, pero el proceso de entrenamiento sigue siendo igual. En esta aplicación se utiliza la configuración de la Figura anterior porque el objetivo es observar el funcionamiento general de una red *backpropagation*, mostrando las bondades de este tipo de red para aproximar funciones (en este caso las curvas de carga de la población en estudio) y su capacidad para predecir comportamientos futuros de patrones no presentados en la etapa de aprendizaje.

Los valores de entrada a la red se agrupan en un vector de dos entradas ***p***, (Tabla de Datos), las cuales describen el día y la hora en que desea saberse el valor de consumo. La red se entrena con valores de salida normalizados porque de esa forma se obtiene una mejor generalización de la red en el proceso de aprendizaje; el proceso de normalización consiste en dividir cada uno de los valores de demanda de la Tabla de Datos por el valor de demanda máximo, de tal forma que el mayor valor de entrada a la red será uno:

$$Demanda\ Normalizada = \frac{Valor\ de\ la\ Demanda}{Demanda\ Máxima}$$

El vector de salida ***t***, constituyen los valores de consumo medido en el día de la semana y hora dados en la Tabla de Datos.

Con esta información, se recomienda:

- Analizar el problema y el archivo de datos correspondiente.
- Preparar los datos de entrada y los de salida.
- Preparar un script de MATLAB para entrenar y simular la red neuronal con la arquitectura sugerida y los datos dados.

- Experimentar con diferentes variantes de la arquitectura en lo que a número de neuronas ocultas se refiere, hasta obtener el mínimo error de predicción de consumo posible.
- Interpretar los resultados obtenidos.
- Presentar comentarios, conclusiones y recomendaciones.

9.9.6 Redes de Base Radial

Las redes neuronales de base radial usualmente requieren más neuronas que las redes de retropropagación, pero se entrenan mucho más rápido que éstas.

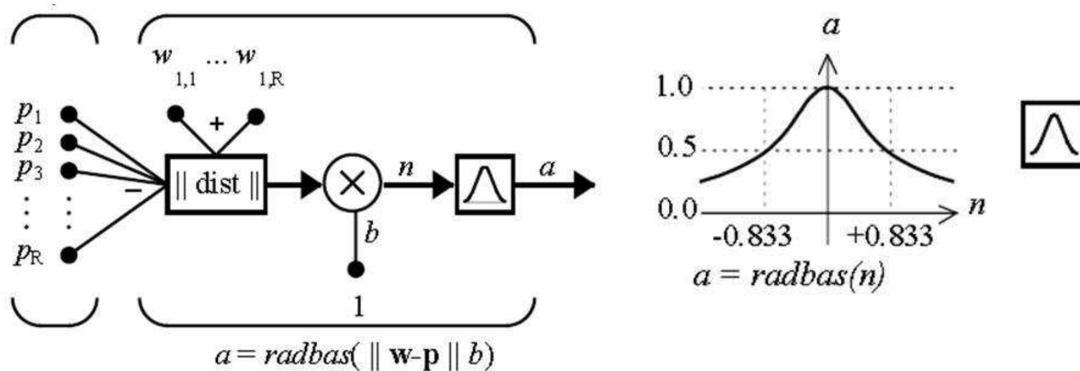


Figura 86. Modelo de Neurona de Base Radial (MATLAB) y Función de Transferencia

El modelo de neurona que utilizan las redes de base radial tiene una función de transferencia gaussiana: $radbas(n) = e^{-n^2}$. Debido la salida tiende a su máximo valor 1, a medida que la distancia entre el vector de pesos $\boldsymbol{w}$ y el vector de entrada $\boldsymbol{p}$ se reduce, la neurona actúa como un detector de identidad. El umbral de decisión b permite ajustar la sensibilidad de la neurona.

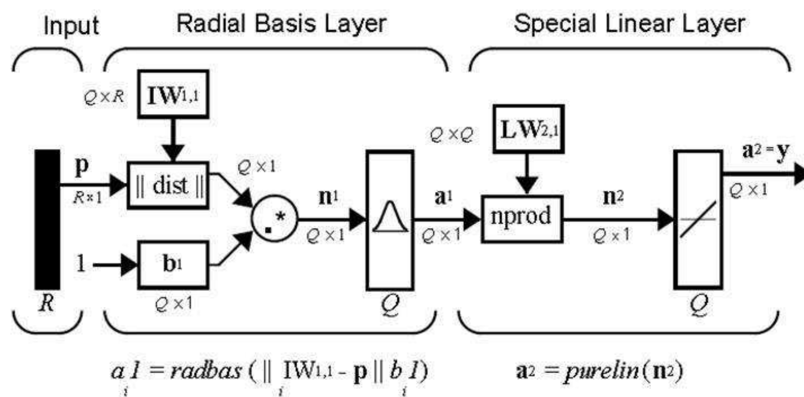
Existen dos variantes:

- Redes neuronales de regresión generalizada, utilizadas para aproximación de funciones.
- Redes neuronales probabilísticas, utilizadas para problemas de clasificación.

Estas redes se diseñan más rápido que las redes de retropropagación, sin embargo, debido a que el número de neuronas de base radial es proporcional al tamaño del espacio de entrada, las redes pueden ser más grandes. Igualmente, pueden ser más lentas para operar ya que realizan más computaciones que otro tipo de redes.

9.9.6.1 Red Neuronal de Regresión Generalizada

Una red neuronal de regresión generalizada se utiliza para aproximación de funciones. Se ha demostrado que, dado un número suficiente de neuronas ocultas, estas redes pueden aproximar una función continua con una precisión arbitraria.



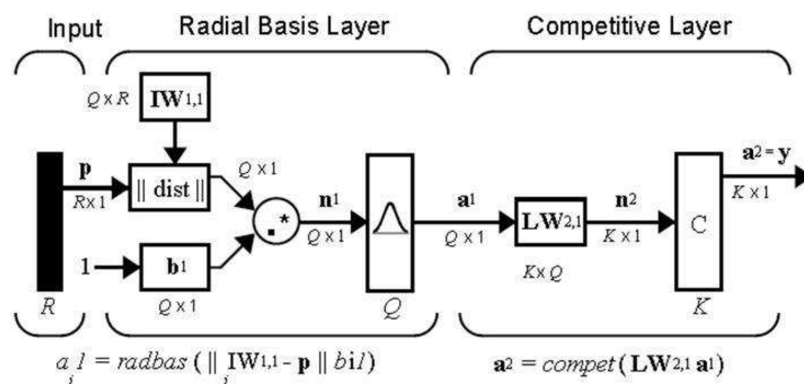
Donde:

- R = N° elementos del vector de entrada
- Q = N° de Neuronas en la capa 1
- Q = N° de neuronas en la capa 2
- Q = N° de pares entrada-salida para entrenamiento

Figura 87. Modelo de Red de Regresión Generalizada (MATLAB)

9.9.6.2 Red Neuronal Probabilística

La red neuronal probabilística, puede ser usada para problemas de clasificación. Cuando se presenta una entrada, la capa de entrada calcula la distancia entre el vector de entrada y los vectores de entrenamiento, produciendo un vector con los valores de diferencia. La segunda capa suma estas contribuciones para cada clase y produce a la salida un vector de probabilidades. Finalmente, la función de transferencias competitiva, a la salida de la segunda capa, escoge la máxima de estas probabilidades y produce un *1* para esta clase y *0* para las demás.



Donde:

- R =N° elementos del vector de entrada
- Q = N° de pares entrada – salida de entrenamiento
- K = N° de clases de los datos de entrada

Figura 88. Modelo de Red Neuronal Probabilística (MATLAB)

9.10 Redes Competitivas y de Auto-organización

La característica de auto-organización es uno de los tópicos más importantes en el campo de las redes neuronales. Este tipo de redes pueden aprender a detectar regularidades y correlaciones en su entrada y adaptar su respuesta futura a estas entradas.

Las neuronas de **redes competitivas** pueden aprender a reconocer grupos de vectores de entrada similares.

Por otra parte, los **mapas de auto-organización**, aprenden a reconocer grupos de vectores de entrada similares de tal manera que neuronas que están físicamente cercanas en la capa, responden a vectores de entrada similares.

9.10.1 Redes Competitivas

Las neuronas en una red competitiva se auto distribuyen la tarea de reconocimiento de vectores de entrada frecuentemente presentados. La arquitectura de una red competitiva se muestra a continuación:

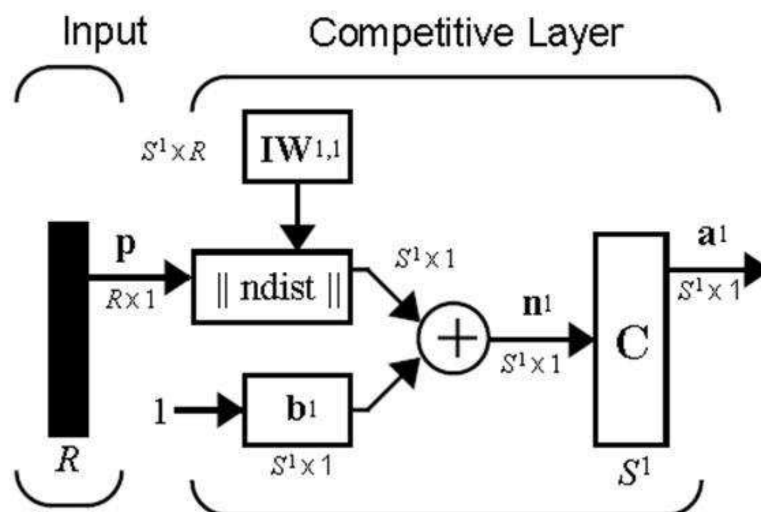


Figura 89. Modelo Arquitectónico de una Red Competitiva (MATLAB)

El bloque $\|ndist\|$ acepta el vector $\boldsymbol{p}$ de entrada y el vector de pesos $\boldsymbol{IW^{1,1}}$ formado a partir de las filas de la matriz de pesos de entrada y entrega un vector con $\boldsymbol{S^1}$ elementos, que son las distancias negativas entre estos vectores.

El vector $\boldsymbol{n^1}$ de entrada a la capa competitiva, es el resultado de la adición de las distancias negativas y los valores de los umbrales de decisión $\boldsymbol{b}$.

La función de transferencia competitiva acepta el vector de entrada $\boldsymbol{n^1}$ y retorna salidas de cero para todas las neuronas, excepto para la ganadora, cuya salida es $\boldsymbol{1}$ y está asociada con el elemento más positivo del vector de entrada $\boldsymbol{n^1}$.

Los pesos de la neurona ganadora (una fila de la matriz de pesos de entrada) son ajustados utilizando la **Regla de Aprendizaje de Kohonen**[183]. Suponiendo que la $\boldsymbol{i}$ neurona gana, la fila $\boldsymbol{i}$ de la matriz de pesos de entrada, se ajusta utilizando la siguiente ecuación:

---

[183] Kohonen, T. *Self-Organization and Associative Memory*, 2nd Edition, Berlin: Springer-Verlag, 1987.

$$iIW^{1,1}(q) = iIW^{1,1}(q-1) + \alpha(p(q) - iIW^{1,1}(q-1))$$

La Regla de Aprendizaje de Kohonen permite que los pesos de una neurona aprendan un vector de entrada. La neurona cuyo vector de pesos está más próximo al vector de entrada, se actualiza para que se acerque mucho más. Debido a esto, estas redes son útiles para aplicaciones de reconocimiento.

9.10.2 Mapas de Auto-Organización

Los denominados *Self Organizing Feature Maps* (**SOFM**), aprenden a clasificar vectores de entrada, de acuerdo a cómo estos están organizados en el espacio de atributos (*feature space*). Se diferencian de las redes competitivas, debido a que neuronas vecinas en el mapa de auto-organización aprenden a reconocer secciones en la vecindad del espacio de entrada. Por lo tanto, las arquitecturas SOFM aprenden tanto la distribución (como las redes competitivas), como la topología de los vectores de entrada en los que se entrenan.

En el mapa de atributos se posicionan más neuronas para reconocer partes del espacio de entrada donde más vectores se encuentran y menos en donde hay escasez. En un mapa de auto-organización se permite que neuronas vecinas a la ganadora a tener valores en la salida. Con esto se consigue una transición más suave de los vectores de salida que lo que se obtiene en redes competitivas donde una sola neurona tiene salida a la vez.

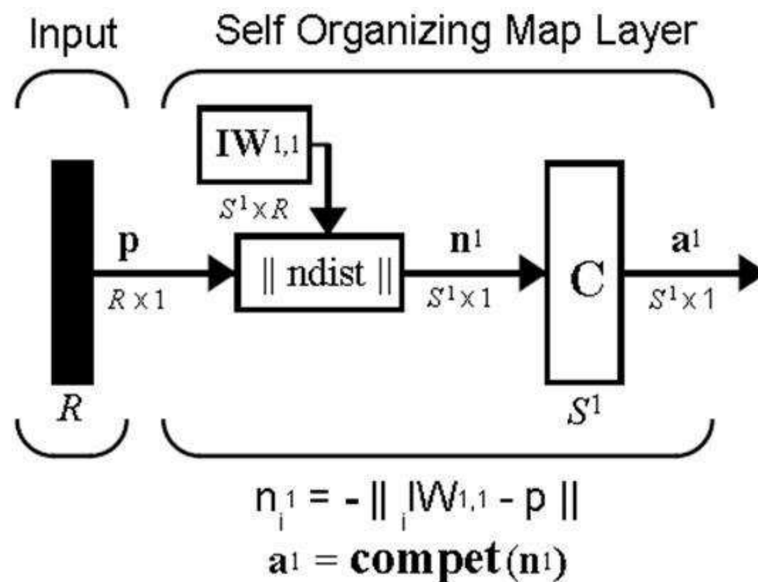


Figura 90. Modelo de una Arquitectura de Mapa de Auto-organización

9.10.3 *Learning Vector Quatization* (LVQ)

La cuantificación de vectores de aprendizaje, es un método para entrenar capas competitivas de manera supervisada. La arquitectura de una red LVQ se muestra a continuación:

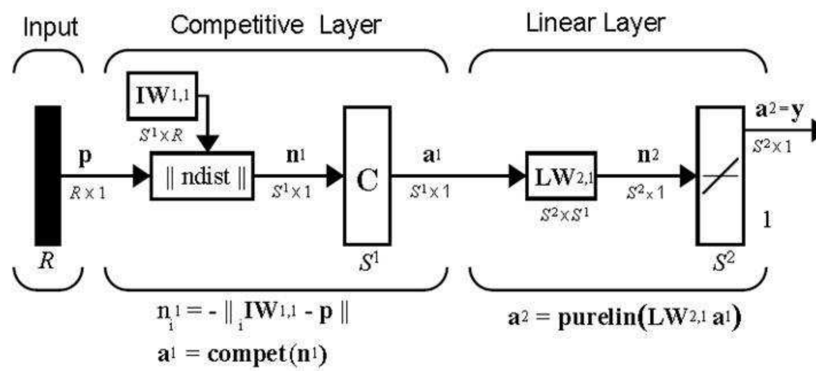


Figura 91. Modelo de Arquitectura de Red LVQ

Donde:

- R = N° de elementos en el vector de entrada
- S^1 = N° de neuronas competitivas.
- S^2 = N° de neuronas lineales

Las redes LVQ utilizan una capa competitiva para encontrar subclases de los vectores de entrada; y, luego los combinan en clases objetivo.

A diferencia de los perceptrones, las redes LVQ pueden clasificar cualquier conjunto de vectores de entrada, lineales o no. El único requerimiento es que la capa competitiva tenga suficientes neuronas y que para cada clase se asigne suficientes neuronas competitivas.

10 CAPÍTULO X: COMPUTACIÓN EVOLUTIVA

10.1 Introducción

Todos los sistemas biológicos son el resultado de procesos evolutivos. La robustez y adaptabilidad de los sistemas biológicos son una fuente de inspiración para el desarrollo de sistemas de HW y SW con características comparables.

La evolución natural no tiene una meta predeterminada y es en esencia un proceso de adaptación abierto; sin embargo, la versión artificial es un proceso de optimización que trata de encontrar soluciones a problemas predefinidos.

La teoría de la evolución natural descansa sobre 4 pilares:

- **Población**.- La premisa de la evolución es la existencia de una población.
- **Diversidad**.- Implica que las características de los individuos varían de una población a otra.
- **Heredad**.- Indica que ciertas características individuales de los padres, pueden transmitirse a los descendientes a través de la reproducción.
- **Selección**.- Establece que sólo una parte de la población es capaz de reproducirse y transmitir sus características a generaciones futuras.

El avance reciente de la genética y de la genómica funcional, ha aportado claves a los mecanismos moleculares y procesos que intervienen en la heredad y la diversidad.

Las especies se crean, evolucionan y desaparecen si no se adaptan de forma que solo los que mejor se adapten al medio sobreviven para perpetuar sus aptitudes. De acuerdo con esta visión de la evolución, la computación ve en este marco un claro proceso de optimización: se toman los individuos mejor adaptados, se cruzan, generando nuevos individuos que contendrán parte del código genético de sus antecesores, y el promedio de adaptación de toda la población se mejora.

Los inicios de la computación evolutiva se remontan a finales de los años 50 con los trabajos de **H-J Bremermann**[184], **R. M. Friedberg**[185,186] y otros, el campo permaneció inactivo por tres décadas debido a la limitada capacidad computacional disponible y deficiencias de los

---

[184] http://turnbull.dcs.st-and.ac.uk/history/Printonly/Bremermann.html
[185] Friedberg, R. M. *A Learning Machine: Part I*. IBM Journal, 2 (1), 1958, pp. 2-13.
[186] Friedberg, R. M., B. Dunham, and J. H. North. *A Learning Machine: Part II*. IBM Journal, 3 (7), 1959, pp. 282-287.

primeros métodos, hasta que la llegada de nuevos aportes de **J. H. Holland**[187], **I. Rechenberg**[188] y **L. J. Fogel**[189] cambiaron el escenario.

La computación evolutiva se ha convertido en un método clave para resolución de problemas difíciles de optimización, debido, entre otros factores, a la flexibilidad y adaptabilidad en la resolución, combinados con la robustez y las ventajas de la búsqueda global.

10.2 Fundamentos de la Computación Evolutiva

La unidad fundamental de información en los seres vivos es el gen. El gen es parte de una estructura denominada cromosoma. El material genético de un individuo se conoce como el **genotipo**. Su manifestación como organismo, que determina o afecta la propiedad visible se denomina **fenotipo**. Los organismos vivos pueden ser visualizados como un dual de su genotipo (la codificación genética) y de su fenotipo (la propiedad visible).

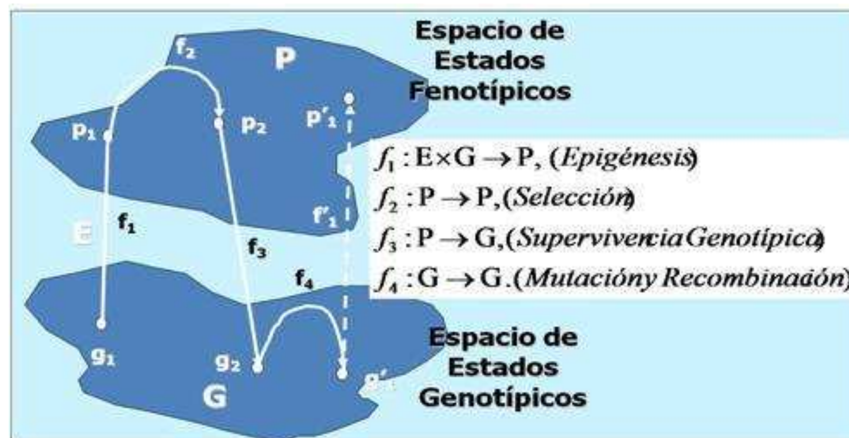


Figura 92. Espacios Genotípicos y Fenotípicos en la Evolución

Un problema de optimización requiere hallar un conjunto de parámetros de forma que se cumpla un cierto criterio de calidad que se quiere optimizar, es decir, maximizando o minimizando una cierta función de evaluación f(x) dada.

La computación evolutiva es especialmente útil cuando se enfrenta problemas difíciles o complejos, como lo son aquellos caracterizados por una alta dimensionalidad, multimodalidad, fuerte no linealidad, no diferenciabilidad, presencia de ruido y cuando se trata con funciones dependientes del tiempo.

---

[187] Holland, John H. *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*. Oxford, England: U Michigan Press. 1975.

[188] Rechenberg, Ingo: *Evolutionsstrategie - Optimierung technischer Systeme nach Prinzipien der biologischen Evolution* (PhD thesis), Technical University of Berlin, 1971.

[189] Fogel, L. J., Owens A. J., and Walsh M. J.. *Artificial Intelligence through Simulated Evolution*. New York: John Wiley, 1966.

La computación evolutiva comprende aplicaciones basadas en los mecanismos de la evolución natural tales como la genética biológica y la selección natural. El campo de la computación evolutiva, en general incluye cuatro paradigmas:

- Algoritmos genéticos.
- Programación evolutiva.
- Estrategias de evolución.
- Programación genética.

Los paradigmas de la computación evolutiva difieren de los métodos tradicionales, en cuatro aspectos fundamentales:

- Trabajan con un código del conjunto de parámetros. Este código es una cadena de caracteres de valores binarios o variables de valor real.
- Realizan la búsqueda de la potencial solución en una población de puntos, no en un sólo punto.
- Utilizan la información de una función objetivo, no derivadas u otro conocimiento auxiliar.
- Usan reglas probabilísticas de transición, en lugar de reglas determinísticas.

10.3 Algoritmos Genéticos

Los Algoritmos Genéticos[190] son métodos adaptativos que se derivan de la simulación de los procesos genéticos naturales. Comprenden los mecanismos de selección (la supervivencia del más apto), cruce (intercambio genético) y mutación. Se utilizan para resolver problemas de búsqueda y optimización.

10.3.1 Características de los Algoritmos Genéticos

Los Algoritmos Genéticos trabajan en el espacio del genotipo del código de la información:

- Se selecciona una población de individuos, cada uno de los cuales representa una solución factible a un problema dado.
- Cada individuo es codificado como una cadena de caracteres binarios (cromosomas), y se le asigna un valor de **aptitud** relacionado con la bondad de la solución que representa.
- Dependiendo de la aptitud de un individuo, la probabilidad de que sea seleccionado para la reproducción y por tanto de que su material genético se propague en sucesivas generaciones, puede ser mayor o menor. Cuanto mayor sea la aptitud de un individuo, mayor será la probabilidad de que el mismo sea **seleccionado** para reproducirse intercambiando aleatoriamente su material genético con otro individuo seleccionado de igual forma.

[190] Holland, John H. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, USA, 1975

- Este **cruce** producirá nuevos individuos, descendientes de los anteriores, los cuales comparten algunas de las características de sus ancestros.
- Ocasionalmente, una variante nueva es probada, cambiando aleatoriamente un gen de algún cromosoma descendiente, como una medida de posible mejora (**mutación**).

```
INICIO /* Algoritmo Genético Simple */
   Generar una población inicial.
   Computar la función de aptitud de cada individuo.
   MIENTRAS NO Terminado HACER
   INICIO /* Producir nueva generación */
      PARA Tamaño Población/2 HACER
      INICIO /* Ciclo Reproductivo */
         Seleccionar 2 individuos de la generación anterior,
         con probabilidad proporcional a la función de aptitud
         de cada individuo.
         Cruzar, con cierta probabilidad, los 2 individuos
            obteniendo 2 descendientes.
         Mutar los 2 descendientes, con cierta probabilidad.
         Calcular la función de aptitud de los 2 descendientes.
         Insertar los dos descendientes en la nueva generación.
      FIN /* Ciclo Reproductivo */
    SI La población ha convergido ENTONCES
       Terminado = VERDADERO
   FIN /* Producir nueva generación */
FIN /* Algoritmo Genético Simple */
```

Figura 93. Pseudocódigo de un Algoritmo Genético Simple

10.3.2 Los Algoritmos Genéticos en Acción

Para ilustrar la operación de los algoritmos genéticos, supóngase que el objetivo es encontrar el máximo número de una serie que va desde 0 hasta 255. Aplicando el pseudocódigo del algoritmo genético simple, se genera aleatoriamente una población inicial de 10 ejemplares pertenecientes a la serie.

10.3.2.1 Codificación de la variable

Para codificar la variable que representa la serie de números entre 0 y 255, se puede utilizar notación binaria, con 8 bits. Esta cadena de 1s y 0s que representa cada número constituye un cromosoma. A cada cromosoma se le asigna un valor de aptitud. En este caso, como el objetivo es localizar el máximo número, se puede utilizar como función de optimización el cuadrado del número decimal ya que, mientras más cerca esté del 255, mayor será su aptitud para constituirse en solución al objetivo esperado.

10.3.2.2 Proceso de Selección

Para iniciar el proceso de selección, se calcula la aptitud relativa de cada ejemplar, en relación a la aptitud total de la población. Con estos valores, se puede construir una ruleta en la que las áreas de los sectores circulares son proporcionales a la aptitud relativa de cada ejemplar.

A continuación se procede a hacer girar la ruleta dos veces, para seleccionar los dos ejemplares más aptos de la población inicial. Nótese que los ejemplares más aptos ocupan

las mayores áreas, por lo que al jugar la ruleta, estos son los que más probabilidad tienen de ser seleccionados. Estos detalles se muestran en las siguientes figuras.

ITEM	DECIMAL	BINARIO	APTITUD	APTITUD RELATIVA
1	127	01111111	16129	11%
2	82	01010010	6724	5%
3	104	01101000	10816	7%
4	69	01000101	4761	3%
5	55	00110111	3025	2%
6	221	11011101	48841	34%
7	16	00010000	256	0%
8	211	11010011	44521	31%
9	27	00011011	729	1%
10	98	01100010	9604	7%
			145406	100%

Figura 94. Población inicial, cromosomas y aptitud

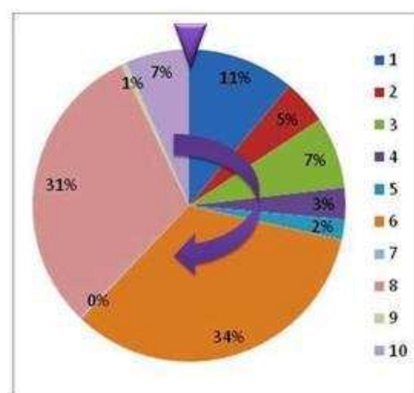


Figura 95. Ruleta de Selección del más Apto

10.3.2.3 Cruce y Mutación

Supóngase que la ruleta selecciona el ítem 8 (`211`) y el ítem 6 (`221`). Aleatoriamente se determina un punto de cruce y se procede a intercambiar el material genético de los dos padres, para configurar sus dos hijos. Hecho esto, a continuación, se introduce una mutación. En este caso, se complementa el valor del gen, escogido aleatoriamente, y se obtienen los dos ejemplares que ingresan a la nueva población. En este ejemplo, aparece el número `251 [11111101]` y se mantiene el `211 [11010011]`.

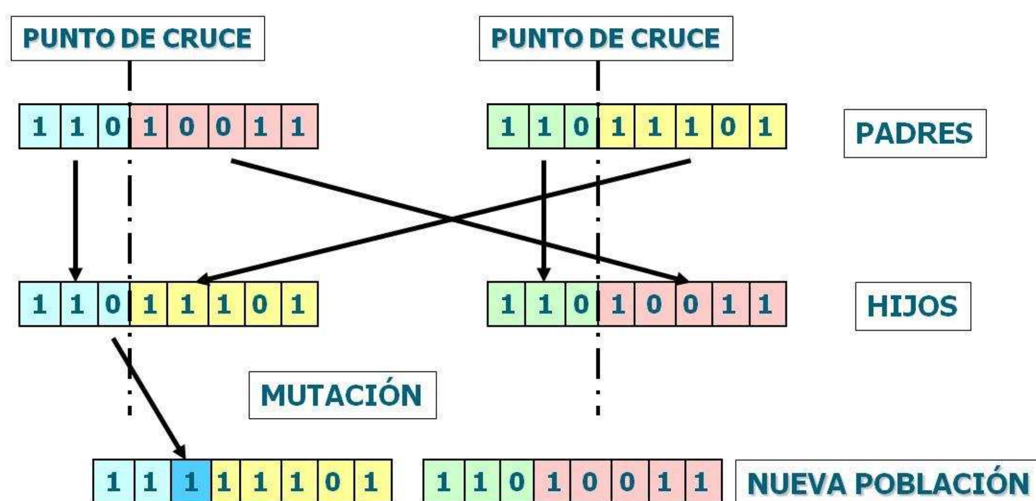


Figura 96. Operaciones de cruce y mutación

Este proceso se realiza hasta emparejar todos los ejemplares de la población inicial, esto es 4 veces más. Hasta que la mayoría de ejemplares se aproximen al número `255`, se produce una nueva generación. Como se puede apreciar, a través de estas operaciones, van apareciendo números de la serie que inicialmente no estuvieron presentes en la población.

Además, generación tras generación, los nuevos ejemplares irán aproximándose al 255, hasta que eventualmente este número domine la población o se agote el material genético y se estanque la evolución, en cuyo caso se tendrá que iniciar un nuevo proceso ya sea con los mismos ejemplares iniciales o generando aleatoriamente una nueva población inicial.

10.3.3 Modificaciones al Algoritmo Genético Simple

Entre las posibles modificaciones al algoritmo genético simple, se pueden considerar las siguientes:

- Utilizar una población variable.- En este caso, regularmente se introduce nuevo material genético para refrescar la población.
- Representar los cromosomas utilizando el Código Gray.- En ciertos problemas puede reducirse las variaciones entre cromosomas de ejemplares sucesivos. El código Gray se caracteriza porque el código binario de dos números sucesivos cambia solo en un bit.

$$b_{m-1}2^{m-1} + b_{m-2}2^{m-2} + \ldots + b_1 2^1 + b_0 2^0 \;(\text{Número Binario})$$

$$\begin{pmatrix} g_i = b_i \oplus b_{i+1} & 0 \le i \le m-2 \\ g_{m-1} = b_{m-1} & \end{pmatrix} \quad (\text{Binario Gray})$$

	Código	
Decimal	Binario	Gray
250	11111010	10000111
251	11111011	10000110
252	11111100	10000010
253	11111101	10000011
254	11111110	10000001
255	11111111	10000000

Figura 97. Código Equivalente Binario – Gray

- Métodos de Selección.- A más de la ruleta, se puede utilizar una selección aleatoria o una selección ranqueada.
- Operaciones Genéticas.- Para problemas en los que las variables se codifican en cromosomas largos, puede ser de ayuda utilizar múltiples puntos de cruce. Otras opciones son: cruce uniforme, máscaras aleatorias o reordenamiento.

10.4 Programación Evolutiva

Fue utilizada por primera vez por **Lawrence J. Fogel** en los EE.UU. en los 1960s con el fin de utilizar la evolución simulada como un proceso de aprendizaje para generar inteligencia artificial. Actualmente, la programación evolutiva es un dialecto de la amplia gama de la computación evolutiva[191].

[191] Fogel, L.J. *Intelligence through Simulated Evolution: Forty Years of Evolutionary Programming*, John Wiley, USA, 1999.

La programación evolutiva se deriva de la simulación del comportamiento adaptativo propio de un proceso evolutivo. Se caracteriza por su énfasis en el desarrollo de modelos de comportamiento. Trabaja en el espacio del fenotipo, es decir, la evolución es en el comportamiento observable.

La programación evolutiva se utiliza usualmente para resolver problemas de predicción. Emplea máquinas de estados finitos para la predicción y la evolución.

- El entorno es descrito como una secuencia de símbolos tomados de un alfabeto finito.
- El algoritmo evolutivo opera sobre la secuencia de símbolos observados, de tal forma que produce un símbolo de salida que maximiza el resultado, tomando en cuenta el siguiente símbolo que aparecerá (predicción) y una función de costo definida que guía el proceso.

Su operador principal es la mutación, los miembros de la población son vistos como parte de una especie específica en lugar de miembros de la misma especie.

```
INICIO  /* Programación Evolutiva: Procedimiento */
   Generar una población inicial.
   MIENTRAS NO Terminado HACER
   INICIO  /* Buscar Solución */
       Exponer la población a su entorno.
       Calcular la aptitud de cada individuo.
       Aleatoriamente mutar cada miembro de la población
           y generar descendientes.
       Calcular la aptitud de los descendientes.
       Seleccionar los miembros más aptos para la nueva
generación.
       SI el objetivo se ha logrado ENTONCES
           Terminado = VERDADERO
   FIN /* Buscar Solución */
FIN /* Programación Evolutiva: Procedimiento */
```

Figura 98. Pseudocódigo de Algoritmo para Programación Evolutiva

10.5 Estrategias de Evolución

Las estrategias de evolución están basadas en el concepto de meta evolución y, al igual que la programación evolutiva, enfatizan el aspecto fenotípico del proceso evolutivo.

A pesar de utilizar las operaciones de mutación y cruce, que en este caso se las denomina operaciones de recombinación, las perspectivas de ellas son diferentes a las utilizadas por los algoritmos genéticos o la programación evolutiva:

- Operaciones de mutación o recombinación que caen fuera de una ventana de evolución predefinida por la función de optimización, no son de utilidad.
- El ajuste dinámico del tamaño de la mutación a una ventana de evolución también dinámica, favorece la meta evolución.

A diferencia de los algoritmos genéticos, las estrategias de evolución manipulan los valores de las variables, durante la recombinación:

- Forma un descendiente utilizando los datos de dos padres seleccionados aleatoriamente.
- El valor de la variable del descendiente, es igual al valor intermedio de los valores de sus padres.

La estrategia de evolución opera con una mayor cantidad de descendientes que los otros paradigmas de computación evolutiva:

El número de descendientes (N) es mayor que el número de padres (M), en una relación típica N/M igual a 7.

```
INICIO  /* Estrategias de Evolución: Procedimiento */
   Generar una población inicial.
   MIENTRAS NO Terminado HACER
   INICIO  /* Buscar Solución */
       Realizar la recombinación de pares de los M padres
           para generar N descendientes (N/M = 7).
       Mutar a todos los descendientes.
       Calcular la aptitud de los N descendientes.
       De entre los N descendientes, seleccionar los M
           miembros más aptos para la nueva generación.
       SI el objetivo se ha logrado ENTONCES
           Terminado = VERDADERO
   FIN /* Buscar Solución */
FIN /* Estrategias de Evolución : Procedimiento */
```

Figura 99. Pseudocódigo de Procedimiento de Estrategias de Evolución

10.6 Programación Genética

La programación genética está diseñada para evolucionar programas de computadora, genéticamente. A diferencia de las tres implementaciones anteriores, que generalmente emplean cromosomas en forma de cadena de caracteres como miembros individuales de la población, la programación genética utiliza árboles para representar a los programas sujetos a evolución:

- Las funciones definidas para el problema aparecen en los nodos (conjunto interno).
- Las variables de estado y las constantes están localizadas en las hojas del árbol (conjunto terminal).

```
INICIO /* Programación Genética: Procedimiento */
   Inicializar la población de programas de computadora.
   MIENTRAS NO Terminado HACER
   INICIO /* Buscar Solución */
      Determinar la aptitud de cada programa individual.
      Aplicar la reproducción de acuerdo con la aptitud y la
         probabilidad de reproducción.
      Realizar el cruce de las subexpresiones.
      SI el objetivo se ha logrado ENTONCES
         Terminado = VERDADERO
   FIN /* Buscar Solución */
FIN /* Programación Genética : Procedimiento */
```

Figura 100. Pseudocódigo para Procedimiento de Programación Genética

Previo a la ejecución de un proceso de programación genética, se deben cumplir los siguientes pasos:

- Especificar el conjunto de variables de estado y constantes (conjunto terminal).
- Especificar el conjunto de funciones.
- Especificar la medida de aptitud.
- Seleccionar los parámetros del sistema de control.
- Especificar las condiciones de finalización del procedimiento.

10.7 Aplicaciones de Computación Evolutiva

La computación evolutiva resulta atractiva para aplicaciones complejas, con poca información específica acerca del problema y valores óptimos difíciles de encontrar. Los algoritmos evolutivos son altamente tolerantes al ruido y robustos.

Durante algún tiempo las aplicaciones de la computación evolutiva estuvieron en los dominios académicos, pero últimamente aplicaciones de escala industrial han ganado atención internacional.

- **CAD System.-** Una de las mayores aplicaciones comerciales fue incluida por General Electric en su sistema de diseño asistido por computador EnGENEous[192]. Este sistema fue diseñado como una herramienta independiente del dominio, combinando un conjunto de algoritmos de optimización tradicionales, con la conveniencia de un sistema experto para especificar restricciones de diseño e información de control, con la perspectiva de optimización global de los algoritmos evolutivos.
- **Sistema Identikit.-** El sistema ***Faceprints***[193] desarrollado por el departamento de psicología de la ***New Mexico State University***, facilita la generación de rostros de

[192] David J. Powell, Siu Shing Tong, Michael M. Skolnick. *EnGENEous domain independent, machine learning for design optimization*. In Proceedings of the third international conference on Genetic algorithms, 1989, pp. 151 – 159.
[193] Caldwell, C. and Johnston, V. S. *Tracking a criminal suspect through face-space with genetic algorithm*. In Proceedings of the Fourth International Conference on Genetic Algorithms, 1991, pp. 416-421.

sospechosos, ayudando a los testigos a evaluar en una escala subjetiva de 0 a 10, veinte rostros generados por el algoritmo evolutivo. La información ingresada por los testigos es procesada genéticamente por el algoritmo para generar nuevos rostros que poco a poco se asemejan a los sospechosos buscados.

- **Aplicaciones Financieras.**- Una compañía en Nuevo México desarrolló un grupo de herramientas para predicción usando series de tiempo y otras para inversiones, en las que los algoritmos genéticos juegan un papel importante[194]. Estructuras en forma de reglas son evolucionadas y comparadas con datos de series de tiempo para predecir si estas crecerán o no. Las reglas son entrenadas con datos financieros reales, utilizando una función objetivo que trata de minimizar el error de la predicción.
- **Diseño de Redes de Telecomunicaciones.**- Los algoritmos evolutivos han sido aplicados para obtener soluciones cuasi-óptimas al problema del árbol probabilístico de mínimo despliegue. Este problema tiene aplicaciones en el diseño de redes de telecomunicaciones y en el diseño de circuito integrados (VLSI)[195].
- **Optimización de Corte de Material.**- Un algoritmo evolutivo de múltiples opciones se utilizó para minimizar el desperdicio de material producido al cortar piezas para manufactura y otros costos de producción[196].
- **Programación de Transporte de Múltiples Productos por un Oleoducto.**- La optimización del transporte de múltiples productos por un oleoducto es un problema de optimización difícil. Existen varias restricciones a considerar como el volumen de los líquidos a transportar para que llene la tubería, el almacenamiento disponible, la demanda de los productos, entre otros. Los algoritmos evolutivos han sido utilizados con éxito en este tipo de aplicaciones[197].

---

[194] Packard, N. A. *A genetic learning algorithm for the analysis of complex data*. Complex Sys. 4, 1990, pp. 573-586.
[195] **http://www.mcs.utulsa.edu/~rogerw/papers/Abuali-Prufer-CSC-94.pdf**
[196] James C. Bean. *A multiple-choice genetic algorithm for a nonlinear cutting stock problem*. ACM Computing in Science and Engineering, Vol. 2 N° 2, March/April 2000, pp 80-83.
[197] **http://portal.acm.org/ft_gateway.cfm?id=298352&type=pdf**

11 CAPÍTULO XI: SISTEMAS DIFUSOS

11.1 Introducción

Los sistemas difusos constituyen una alternativa a las nociones tradicionales de teoría de conjuntos y la lógica que tiene sus orígenes en la antigua filosofía griega. Actualmente forma parte de las aplicaciones que están a la vanguardia de la Inteligencia Artificial. Sin embargo, a pesar de su remoto origen, es un campo relativamente nuevo y abierto a la investigación.

La precisión de las matemáticas debe su éxito, en parte a los esfuerzos de Aristóteles y otros filósofos que lo precedieron. En su esfuerzo por establecer una teoría concisa de la lógica y luego de las matemáticas, propuso las **Leyes del Pensamiento**[198]:

- Ley de identidad
- Ley de la no contradicción
- Ley de la exclusión del medio

La última de ellas establece que toda proposición debe ser ya sea verdadera o falsa. Lo que en esa misma época era motivo de controversia. En este sentido, Platón sentó las bases para lo que sería la lógica difusa. Propuso que hay una tercera región (más allá de verdadero y falso), donde estos opuestos colapsan. Otros filósofos, modernos se hicieron eco de sus pensamientos, sobre todo Hegel, Marx y Engels. Pero **J. Lukasiewicz**[199] fue el primero que propuso una alternativa sistemática a la lógica bi-valorada de Aristóteles.

En el 1900, Lukasiewicz describe una lógica de tres valores, junto con las matemáticas que lo acompañan. El tercer valor que propone se lo puede traducir mejor como el término posible, y le asignó un valor numérico entre verdadero y falso. Más tarde, declaró que en principio no había nada para evitar la derivación de una lógica de valor infinito.

Pero no fue sino hasta 1965, con el trabajo publicado por **Lotfi A. Zadeh**, que se concretó la noción de una lógica infinitamente valuada. En su obra ***Fuzzy Sets***[200], describe las matemáticas de la teoría de los conjuntos difusos; y, por extensión, de la lógica difusa. Esta teoría propone una función de membresía definida sobre el rango de valores reales [0,0 a 1,0], nuevas operaciones para el cálculo lógico; y, demostró que, en principio, constituye una generalización de la lógica clásica de Aristóteles.

---

[198] Boole, George. *An Investigation of the Laws of Thought on Which are Founded the Mathematical Theories of Logic and Probabilities.* Macmillan, 1854.
[199] Lejewski, C. *Jan Lukasiewicz.* Encyclopedia of Philosophy, Vol. 5, MacMillan, NY: 1967, pp. 104-107.
[200] Zadeh, L.A. *Fuzzy Sets*. Information & Control., Vol. 8, 1965, pp. 338-353.

La idea central de un sistema difuso es que los valores de pertenencia (en **conjuntos difusos**) o valores de verdad (en la **lógica difusa**) se indican con un valor en el rango [0,0 a 1,0], donde 0,0 representa absoluta falsedad y 1,0 que representa la verdad absoluta.

11.2 Teoría de los Conjuntos Difusos

La Teoría de Conjuntos Difusos se refiere al estudio de clases de objetos con fronteras indefinidas, de tal forma que las transiciones entre sus conjuntos son graduales antes que abruptas. Su estudio incluye:

- Lógica difusa
- Aritmética difusa
- Programación matemática difusa
- Topología difusa
- Teoría de gráficos difusos
- Análisis de datos difusos

Lotfi Zadeh, a través de la Teoría de los Conjuntos Difusos propuso una representación de tipo lingüística antes que matemática para los sistemas difusos.

Los conjuntos difusos permiten representar conceptos imprecisos. Están basados en la premisa que los elementos clave en el pensamiento humano no son solamente números, sino que pueden ser representados, aproximadamente, por clases de objetos cuyas **funciones de membresía** pueden tener transiciones graduales entre ellas y traslaparse.

La **Función de Membresía,** define cómo un punto en el espacio de entrada (universo de lectura), es proyectado a un valor de membresía (**difusidad**) comprendido entre 0 y 1. La función de membresía puede ser lineal por tramos o una curva continua de forma arbitraria, escogida de tal manera que satisfaga la naturaleza del problema que representa, desde el punto de vista de simplicidad, conveniencia, rapidez y eficiencia.

La **difusidad** describe el grado de pertenencia o membresía de un elemento a un conjunto difuso. Es una medida de cuán bien una instancia (valor) conforma con un ideal semántico (concepto).

11.3 Conjuntos Difusos

Los conjuntos difusos relacionan un valor que representa un miembro de un conjunto, a un número entre 0 y 1 que indica el grado de pertenencia (**μ**) de ese miembro a dicho conjunto.

Un grado igual a 0 significa que el valor no está contenido en el conjunto, mientras que un grado igual a 1 implica que el valor es totalmente representativo del conjunto.

Estos dos extremos están unidos por una función continua que representa una transición gradual entre 0 y 1. Las siguientes Figuras muestran ejemplos de conjuntos difusos con una función continua y con funciones lineales por tramos.

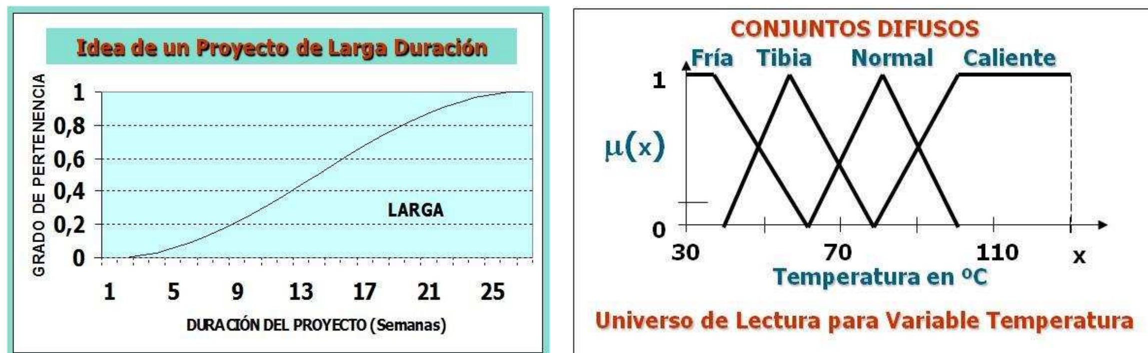


Figura 101. Ejemplos de Conjuntos Difusos

Un conjunto difuso tiene varias propiedades intrínsecas que afectan la forma cómo es utilizado y cómo participa en un modelo:

- La **función** que describe la membresía de los elementos del conjunto difuso (lineal, triangular, trapezoidal, sigmoidal, gaussiana, ...).
- La **dimensión vertical** (altura, normalización).
- La **dimensión horizontal** (dominio, umbral, conjunto válido y universo de lectura)
- La **variable lingüística**, que puede tener uno o más **valores lingüísticos**. Estos corresponden a los nombres de las funciones de membresía que representan los conjuntos difusos.

Los valores lingüísticos, pueden ser vistos como una forma de compresión de datos, que se conoce como nivel de granularidad.

11.4 Operaciones Difusas

Estas operaciones difusas son una generalización de las que se aplican a los conjuntos precisos o clásicos. Las más utilizadas son las denominadas operaciones estándar:

- **Unión**:
  A ∪ B = Max(μA[x], μB[x])
- **Intersección**:
  A ∩ B = Min(μA[x], μB[x])
- **Complemento**:
  ~A = 1- μA[x]
- **Normalización**:
  Norm(A) = μA[x]/ Max(μA[x])

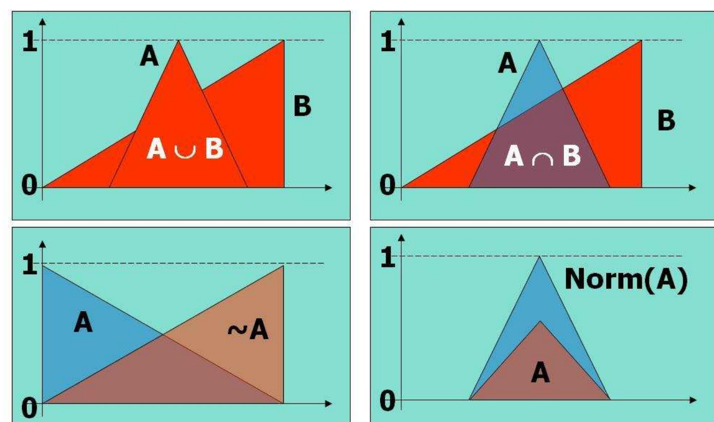


Figura 102. Operaciones difusas estándar

Otros operadores básicos, no lineales que se aplican a los conjuntos difusos son los siguientes:

- **Dilatación:**

  $Dil(A) = (\mu_A[x])^{1/2}$

- **Concentración:**

  $Con(A) = (\mu_A[x])^2$

- **Intensificación:**

$$Int(A) = \begin{cases} 2*(\mu_A[x])^2 & 0.0 < \mu_A[x] \leq 0.5 \\ 1 - 2*(1 - \mu_A[x])^2 & 0.5 < \mu_A[x] \leq 1.0 \end{cases}$$

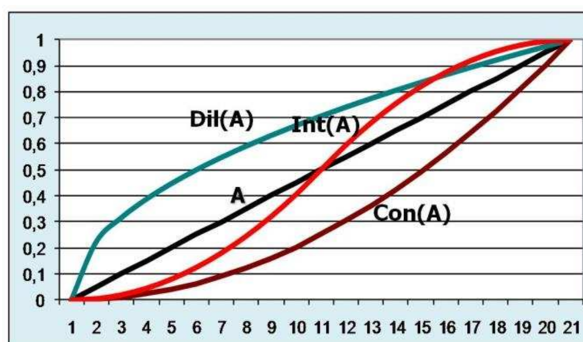


Figura 103. Operadores difusos básicos no lineales

11.5 Calificadores Difusos

Las variables lingüísticas pueden tener asociados calificadores difusos. Actúan de igual manera que los adjetivos o adverbios modificando el significado de los sustantivos**:** muy, bastante, más o menos, algo, casi, alrededor de, definitivamente, positivamente, ligeramente, extremadamente, ...

Los calificadores difusos cambian la forma de la función de membresía, son de naturaleza heurística y la definición de sus funciones es un tanto arbitraria. En la Figura siguiente se da un conjunto de calificadores y su posible operación que representa su semántica.

Calificador	OPERACIÓN
A y B	$A \cap B$
A o B	$A \cup B$
No (A)	~A
Muy (A)	Con(A)
Ligeramente(A)	Dil(A)
Más o Menos (A)	Norm(Int(Dil(A)) ∩ No(A))
Razonablemente (A)	Con(Norm(Dil(Con(A) ∩ Int(Con(A)))
Cualquier Cosa Menos (A)	Int(No(A))
Algo Como (A)	Int(Dil(A))
Parecido a (A)	Norm(Int(Dil(A)) ∩ Int(Dil(No(A)))
En Cierto Sentido (A)	Norm(Int(A) ∩ No(A))

Figura 104. Calificadores y operaciones difusas

11.6 Lógica Difusa

La lógica difusa, es la base para el desarrollo de una nueva tecnología en el diseño de sistemas de inteligencia artificial embebidos en las áreas de procesos e ingeniería de control, con el beneficio de:

- Reducción significativa del tiempo de desarrollo.
- Modelación de sistemas no-lineales muy complejos.
- Implantación de controles utilizando sensores y circuitos integrados más baratos.
- Diseño e instalación de sistemas de control avanzados empleando ingenieros antes que científicos.

La Lógica Difusa proporciona los medios para reducir y explicar la complejidad de sistemas:

- Mucho de la complejidad está relacionada con la forma en que las variables del sistema son representadas y procesadas.
- Los humanos no razonamos en términos de símbolos discretos y números, sino en términos lógicos ambiguos, inciertos, inexactos e imprecisos.

La Lógica Difusa es un cálculo de compatibilidad. Trata de describir las características de propiedades que tienen valores que varían en forma continua asociando particiones de estos valores con un membrete semántico.

Mucho del poder descriptivo de la lógica difusa está asociado con el hecho de que esas particiones semánticas pueden traslaparse.

11.7 Sistemas Difusos

Un sistema difuso está conformado por un conjunto de reglas difusas que convierten entradas a salidas. Una regla difusa tiene la forma:

$$\text{Si } X \text{ es } A \text{ entonces } Y \text{ es } B$$
$$\text{Donde } A \text{ y } B \text{ son conjuntos difusos}$$

Los conjuntos difusos son los bloques constructivos de las reglas difusas. En consecuencia, un sistema difuso se puede considerar como una proyección o función que transfiere alternativas de un espacio de entrada a resultados en un espacio de salida. Basada en esta consideración, la ingeniería de un sistema difuso comprende los siguientes pasos:

1. Escoger las variables de entrada y de salida (variables lingüísticas)
2. Escoger los valores lingüísticos (conjuntos difusos) para las variables de entrada y salida
3. Relacionar los conjuntos de salida a los de entrada, a través de reglas difusas.

A manera de ejemplo considérese un sistema de control de temperatura ambiental que utiliza ventiladores. La variable de entrada X, es la temperatura ambiental y la de salida Y

es la velocidad de los ventiladores. Para la temperatura del ambiente se escogen como valores lingüísticos: fría, fresca, normal, abrigada, caliente. Para la velocidad de los ventiladores se escoge: mínima, baja, media, alta, máxima.

Las reglas difusas correspondientes son:

- Regla 1: Si la temperatura ambiente está fría entonces la velocidad de los ventiladores es mínima.
- Regla 2: Si la temperatura ambiente está fresca entonces la velocidad de los ventiladores es baja.
- Regla 3: Si la temperatura ambiente está normal entonces la velocidad de los ventiladores es media.
- Regla 4: Si la temperatura ambiente está abrigada entonces la velocidad de los ventiladores es alta.
- Regla 5: Si la temperatura ambiente está caliente entonces la velocidad de los ventiladores es máxima.

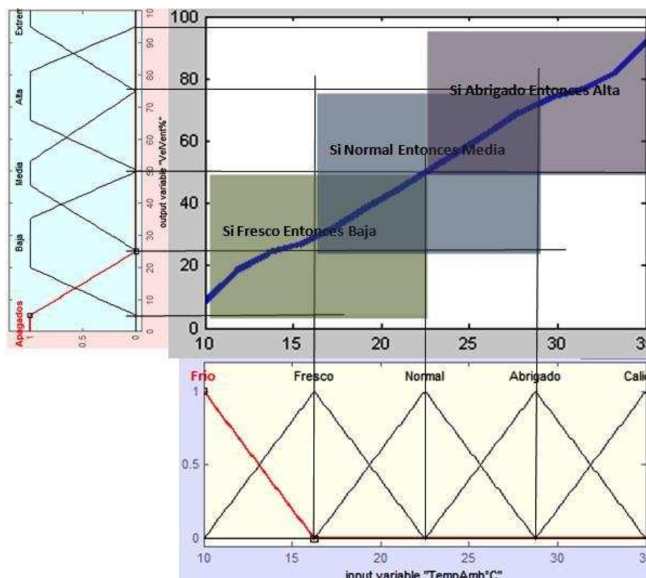


Figura 105. Función de transferencia del sistema difuso

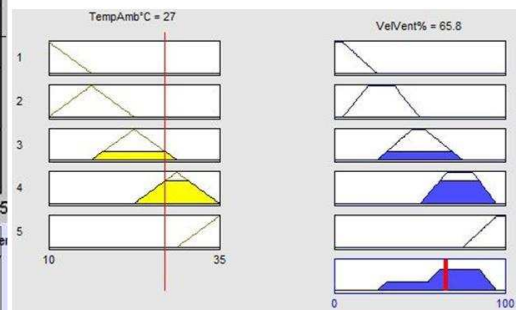


Figura 106. Operación de las reglas difusas

Las figuras anteriores muestran la solución de este ejemplo realizada en el ***Fuzzy Logic Toolbox*** de MATLAB. En el visor de operación de las reglas difusas se ve que para una temperatura ambiente de 24°C los ventiladores deben estar a una velocidad del 65,8% de su velocidad máxima. La interpretación de las reglas difusas se realiza en tres pasos:

- **Fusificar las entradas.-** Resolver todas las proposiciones difusas de los antecedentes a un grado de membresía entre 0 y 1. Si sólo hay una parte en el antecedente, entonces este el grado de contribución de la regla.
- **Aplicar operadores fuzzy a antecedentes múltiples.-** Si en el antecedente hay múltiples partes, entonces aplicar los operadores difusos y resolver el antecedente a un solo valor entre 0 y 1.

- **Aplicar el método de implicación.-** Usar el grado de contribución de la regla para conformar el conjunto difuso de salida. Si el antecedente es parcialmente verdadero el conjunto difuso de salida es truncado, de acuerdo al método de implicación utilizado.

En los sistemas difusos, las reglas se ejecutan en paralelo y todas contribuyen al resultado, según su grado de veracidad. La salida de cada regla es un conjunto difuso. Los conjuntos difusos resultantes de todas las reglas son agregados en un solo conjunto difuso de salida. Finalmente, el conjunto resultante es defusificado, es decir resuelto a un valor numérico único.

11.8 Razonamiento Difuso e Inferencia

El razonamiento difuso es un método que interpreta los valores de un vector de entrada y, basado en un conjunto de reglas, asigna valores al vector de salida. Los tres conceptos que constituyen la base del razonamiento difuso, son:

- **Método de Implicación**: Enlace funcional entre el grado de veracidad en las regiones difusas relacionadas por las proposiciones.
    - Mínimo: Valor lingüístico de salida truncado
    - Producto: Valor lingüístico de salida escalado
- **Método de Agregación**: Forma de combinar las regiones difusas relacionadas.
    - Máximo: $Max(A,B,C)= Max(\mu_A[x],\mu_B[x], \mu_c[x])$
    - Suma: $R=\Sigma(\mu_A[x], \mu_B[x], \mu_C[x], ...)$
    - OR Probabilístico: $ProbOR(A, B)=\mu_A[x]+\mu_B[x]-\mu_A[x].\mu_B[x]$
- **Método de Defusificación**: Operación que recibe el resultado de la agregación y entrega un valor resultante esperado.
    - Centroide.
    - Bisector.
    - Media de Valores Máximos.
    - Mayor de los Máximos.
    - Menor de los Máximos.

El proceso de inferencia difusa involucra funciones de membresía, operadores difusos y reglas difusas para relacionar el vector de entrada dado al vector de salida. Estas relaciones proporcionan la base para tomar decisiones o identificar patrones. Hay dos tipos de sistemas de inferencia difusa:

- El propuesto por **Ebrahim Mamdani**; y,
- El propuesto por **Takagi**, **Sugeno** y **Kang**.

11.8.1 Método de Mamdani

El Método de Mamdani es el más común en la metodología de desarrollo de sistemas difusos. El método Mamdani fue de los primeros utilizados para sistemas de control basados utilizando la teoría de los conjuntos difusos.

Fue propuesto en 1975 por Ebrahim Mamdani[201], para controlar una combinación de máquina de vapor y caldera, sintetizando un conjunto de reglas lingüísticas de control obtenidas de experimentados operadores humanos. Su trabajo se basó en una publicación de Lotfy Zadhe acerca de la aplicación de algoritmos difusos en sistemas complejos y procesos de decisión[202]. Se caracteriza por:

- Ser muy intuitivo;
- Tener amplia aceptación; y,
- Ser adecuado para uso humano.

A continuación se presenta un ejemplo en el que se trata de encontrar el precio de un artículo como porcentaje del precio recomendado de venta, en función de la oferta y demanda también estimadas en términos porcentuales relativos a lo que se considera normal en el mercado analizado.

Para oferta y demanda se escogen tres conjuntos difusos: baja, normal y alta. Para el precio se escogen también tres conjuntos difusos: reducido, nominal e incrementado. En la siguiente Figura se muestra la estructura del sistema difuso.

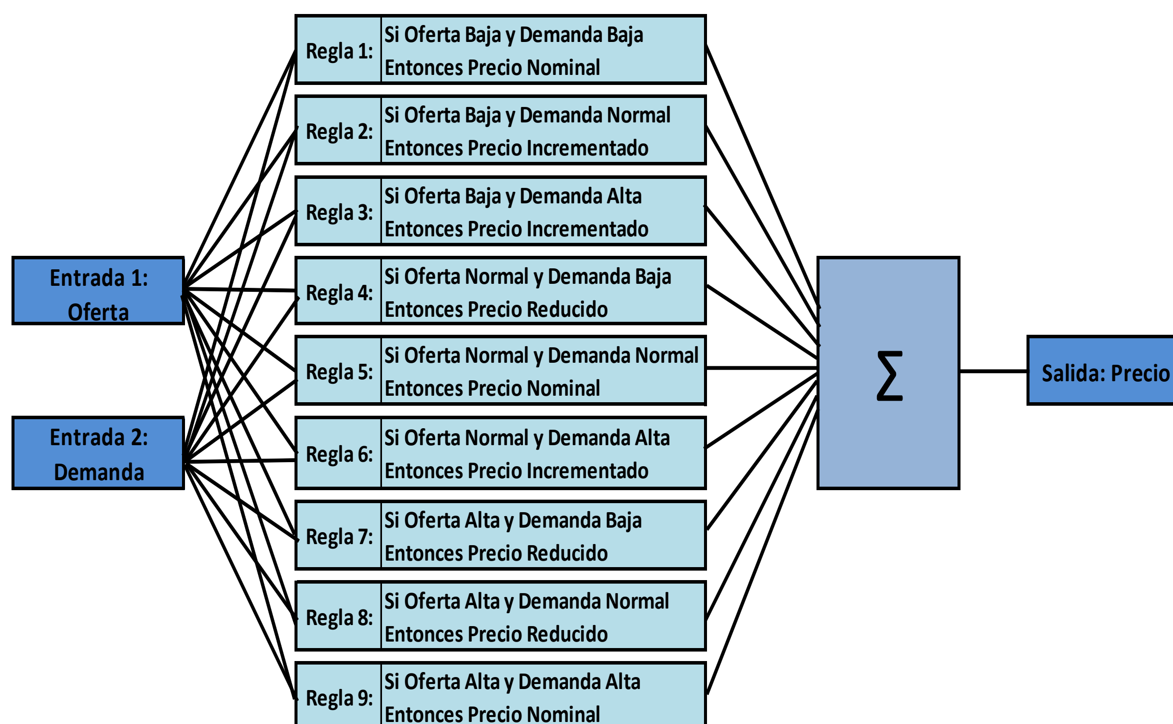


Figura 107. Estructura del sistema difuso `Precio=f(Oferta,Demanda)`

[201] Mamdani, E. H.; Assilian, S. *An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller*. International Journal of Man-Machine Studies, 7, 1, Jan 1975, pp. 1-15,

[202] Zadeh, L. A. *Outline of a new approach to the analysis of complex systems and decision processes*. IEEE Transactions on Systems, Man and Cybernetics, Vol N°3, N° 1, Jan 1973, pp. 28-44.

Con la ayuda del *Fuzzy Logic Toolbox* del MATLAB, se diseña el sistema. Se ajustan los conjuntos difusos de las entradas y la salida hasta que la función de transferencia refleja la respuesta deseada y se comprueba su consistencia con la situación del mundo real. Los resultados se muestran en la siguiente Figura.

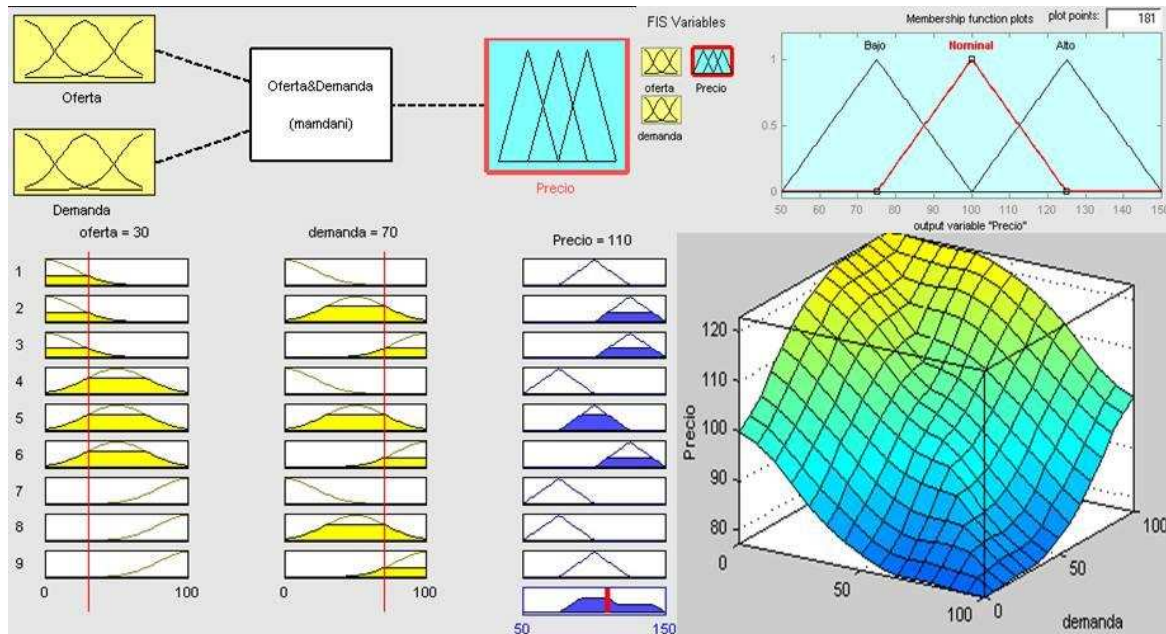


Figura 108. ***Precio=f(Oferta,Demanda)***: Método Mamdani (*Fuzzy Logic Toolbox*, MATLAB)

11.8.2 Método de Sugeno

Fue introducido por Takagi y Sugeno[203] en 1985. Es similar en varios aspectos al método propuesto por Mamdani, excepto porque las funciones de membresía de los valores lingüísticos de las variables de salida tienen forma de espiga (*singleton spikes*). Este método se caracteriza por:

- Ser computacionalmente más eficiente
- Tener superficie de salida continua, garantizada
- Trabajar muy bien con técnicas de control lineal (PID), optimización y adaptación
- Ser adecuado para el análisis matemático.

En la siguiente Figura se muestra el sistema resultante aplicando el método Sugeno a los datos del problema del precio en función de la oferta y demanda. Como se puede apreciar, los resultados son similares.

---

[203] T.Takagi and M.Sugeno, *Fuzzy Identification on Systems and its Applications to Modeling and Control*, IEEE Transactions on Systems, Man, and Cybernetics, Vol.15, 1985, pp.116-132

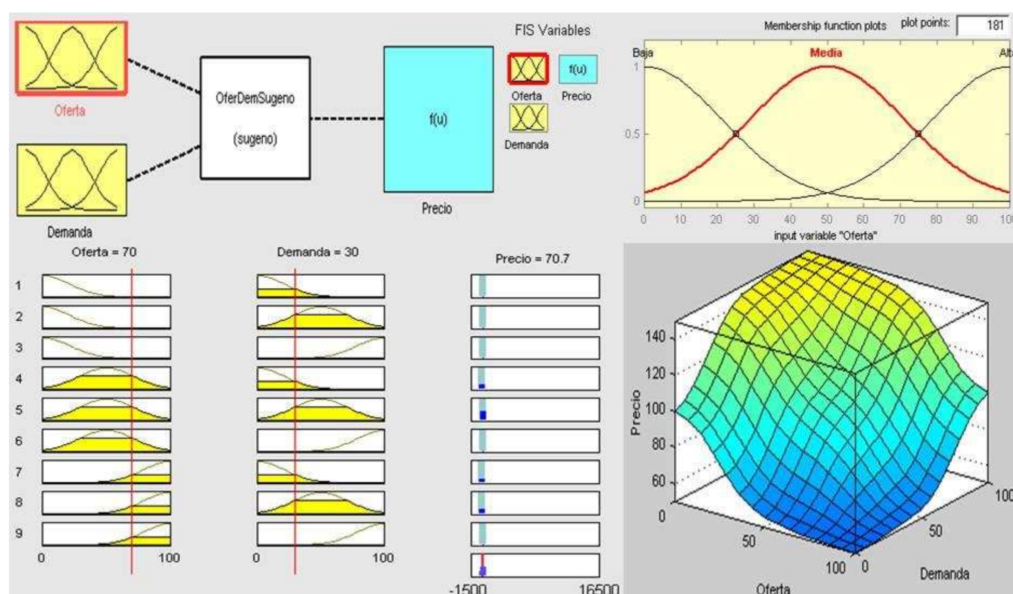


Figura 109. ***Precio=f(Oferta,Demanda)***: Método Sugeno (*Fuzzy Logic Toolbox*, MATLAB)

11.9 Aplicaciones de Sistemas Difusos

Las aplicaciones de los sistemas difusos abarcan múltiples campos del conocimiento, entre ellas se pueden mencionar: Medicina, Robótica, Educación, Transporte, Agricultura, Computación, Automovilismo, Telecomunicaciones, Seguridad / Mantenimiento, Electrónica para el Consumidor, Administración y Gestión Empresarial. En ingeniería: Financiera, Eléctrica, Química, Mecánica, Civil, Ambiental, Geofísica, Nuclear, Aeroespacial y otros.

En particular, las categorías y áreas de aplicación más populares se indican en la siguiente Figura:

Categoría	Ejemplos de Áreas de Aplicación
Control	Es la categoría en la que mayormente se desarrollan aplicaciones
Reconocimiento de Patrones	Imagen (OCR), audio y procesamiento de señales
Análisis Cuantitativo	Investigación de operaciones, estadística y administración
Inferencia	Sistemas expertos para diagnóstico, planeación y predicción; procesamiento de lenguaje natural, interfaces inteligentes, robots inteligentes, ingeniería de software
Recuperación de Información	Bases de datos

Figura 110. Categorías y áreas de aplicación de los Sistemas Difusos.

12 CAPÍTULO XII: AGENTES INTELIGENTES

12.1 Introducción

El término agente, de acuerdo con el diccionario *Merrian-Webster*[204], se define como:

- *Algo que actúa o ejerce poder*
- *Algo que produce o es capaz de producir un efecto*
- *Un medio o un instrumento a través del cual una inteligencia puede alcanzar un resultado*
- *Alguien autorizado para actuar por o en lugar de otro*
- *Una aplicación de computadora diseñada para automatizar ciertas tareas (como colectar información en línea)*

En el entorno de la comunidad de investigadores, agente se define como:

- *Cualquier entidad capaz de percibir su entorno por medio de sensores y ejercer acciones en ese entorno a través de sus actuadores*[205].
- *Término utilizado para representar dos conceptos ortogonales: la habilidad para ejecución autónoma; y, la habilidad para realizar razonamiento en un determinado dominio*[206].
- *Sistema computacional que actúa en un determinado entorno complejo y dinámico, sensa y actúa autónomamente para lograr un conjunto de metas o tareas, para las cuales fue diseñado*[207].
- *Proceso computacional que implementa una aplicación funcional de comunicación autónoma*[208].
- *Entidad persistente de software dedicada a un propósito específico. La persistencia los distingue de las subrutinas, los agentes saben cómo realizar sus tareas, tienen sus propias agendas*[209].
- *Programa que continuamente ejecuta tres funciones: percepción de las condiciones dinámicas en el entorno; acción para afectar las condiciones en el entorno; y,*

---

204 **http://www.merriam-webster.com/dictionary/agent**
205 Russell, S., and Norvig, P. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Inc. USA, 1995.
206 http://www.uv.mx/aguerra/documents/2013-ia2-01.pdf
207 Maes, Pattie. *Artificial Life Meets Entertainment: Life like Autonomous Agents*. Communications of the ACM, 38, 11, 1995, pp. 108-114
208 Foundation for Intelligent Physical Agents, **www.fipa.org**
209 Smith, D. C., A. Cypher and J. Spohrer. *KidSim: Programming Agents Without a Programming Language*. Communications of the ACM, 37, 7, 1994, pp. 55-67

razonamiento para interpretar sus percepciones, resolver problemas, realizar inferencias y determinar acciones[210].

- *Es un sistema computarizado situado en un entorno, en el cual es capaz de ejercer acción autónoma a fin de lograr sus objetivos para los que fue diseñado*[211].

Estas definiciones reflejan importantes características de los agentes, tales como: autonomía, adaptación, reacción, movilidad, cooperación, interactividad, identidad única, pro actividad, sociabilidad y delegación.

12.2 Agentes Racionales

Si se concibe un agente como un sistema cognitivo, capaz de percibir y actuar en un determinado entorno, de manera tal que pueda lograr los objetivos deseados, con base en ciertos supuestos o restricciones, se puede decir que su operación está guiada por la racionalidad.

> ***Principio de Racionalidad***
>
> ***Razonamiento lógico – analítico aplicado a través de una acción comunicativa, para alcanzar alguna meta deseada, logrando acuerdos con otros elementos en un determinado entorno.***

El enfoque del agente racional:

- Es más general que el enfoque de las leyes del pensamiento.
- Es más afín a la manera como se ha producido el avance científico que los enfoques basados en la conducta o el pensamiento humanos.
- Define claramente la norma de la racionalidad, norma que es de aplicación general.

Un agente racional percibe su entorno mediante sensores y responde o actúa por medio de efectores o actuadores. La racionalidad de la acción depende de 4 factores:

- Del conocimiento que el agente posea del entorno en donde opera.
- De la historia perceptual del agente (secuencia de percepciones).
- De las acciones que el agente pueda emprender.
- De la medida con la que se evalúa el éxito logrado.

Un Agente Racional Ideal, es aquel que en todos los casos de posibles secuencias de percepciones, puede emprender todas aquellas acciones que favorezcan obtener el

---

[210] Hayes-Roth, B. *An Architecture for Adaptive Intelligent Systems*. Artificial Intelligence: Special Issue on Agents and Interactivity, 72, 1995, pp. 329-365.

[211] http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.66.6770&rep=rep1&type=pdf

máximo de su medida de rendimiento, basándose en las evidencias aportadas por la secuencia de percepciones y en todo el conocimiento que tiene incorporado.

La conducta de un agente se basa tanto en su propia experiencia como en el conocimiento integrado que le dio la habilidad de operar en un ambiente específico. Un agente será autónomo en la medida en que su conducta esté definida por su propia experiencia.

El objetivo de la IA es el diseño e implementación de un programa de agente que actúe con racionalidad. La estructura de un programa de agente inteligente depende de varios factores:

- Entorno en el que va a trabajar y características de los involucrados en su operación.
- Percepciones, objetivos, metas y acciones que intervienen.
- Medidas de desempeño para evaluar las acciones que pueda emprender el agente.
- Tecnología informática necesaria para su desarrollo, instalación, operación y mantenimiento.

Previo al diseño de un programa de agente, se debe tener una buena idea acerca del entorno en el que va a operar, los involucrados, las percepciones, los objetivos a lograr y las acciones que debe ser capaz de emprender. La siguiente Figura muestra algunos tipos de agentes y las características mencionadas.

TIPO DE AGENTE	ENTORNO	INVOLUCRADOS	PERCEPCIONES	OBJETIVOS	ACCIONES
Sistema de Diagnóstico Médico	Consultorio, hospital, clínica	Ing. Cognitivo Médico Paciente	Síntomas, evidencias, respuestas del paciente	Paciente saludable, reducción de costos	Preguntas, diagnóstico, tratamiento
Sistema Análisis de Imágenes Satelitales	Oficina, Laboratorio	Ing. Cognitivo Ing. Geógrafo	Imágenes satelitales	Identificación correcta de áreas geográficas	Clasificar áreas de la escena
Robot Clasificador de Partes	Banda transportadora de partes	Ing. Cognitivo Ing. Control Operador	Imágenes B/N o a color	Analizar partes e identificarlas correctamente	Colocar las partes en el lugar correcto
Controlador de una Refinería	Refinería de crudo	Ing. Cognitivo Ing. Control Operador	Lecturas de variables del proceso	Lograr pureza, rendimiento, calidad y seguridad	Accionar dispositivos de control del proceso
Asesor Interactivo de Idiomas	Aula Laboratorio	Ing. Cognitivo Profesor de Idiomas Estudiantes	Lenguaje escrito y hablado	Aprendizaje del estudiante	Ejercicios, sugerencias, correcciones, evaluación

Figura 111. Ejemplos de Agentes

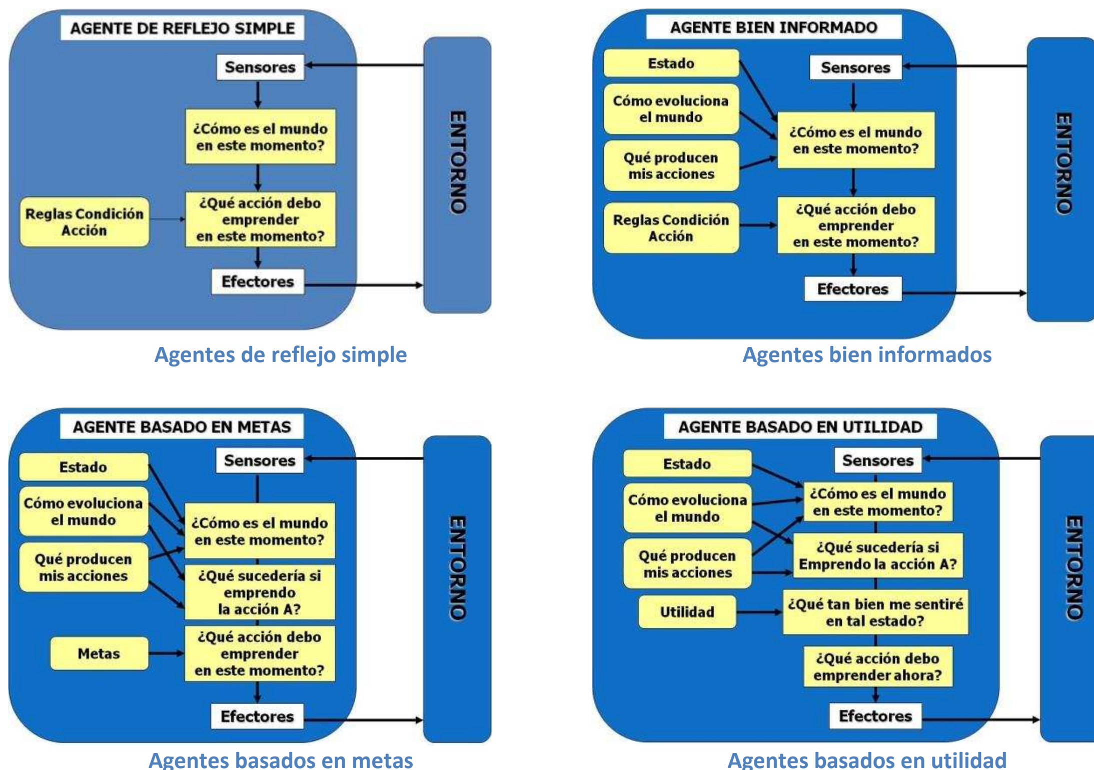
12.3 Tipos de Agentes

Partiendo de las definiciones se pueden identificar las siguientes propiedades que caracterizan a los agentes[212]:

Propiedades	Otros Nombres	Significado
Reactivo	Sensa y Actúa	Responde a cambios en el entorno
Autónomo		Ejercita control sobre sus acciones
Orientado por las Metas	Pro activo, Con propósito	Capacidad de razonamiento
Continuo en el Tiempo		Proceso que se ejecuta continuamente
Comunicativo	Capacidad de Socialización	Se comunica con otros agentes o personas
Aprende	Adaptativo	Cambia su comportamiento según experiencias previas
Móvil		Capaz de moverse de una máquina a otra
Flexible		Las acciones no están prescritas
Carácter		Posee personalidad y estado emotivo

Figura 112. Propiedades de Agentes

Los agentes pueden ser clasificados dependiendo del conjunto de propiedades que tienen:



Agentes de reflejo simple — Agentes bien informados

Agentes basados en metas — Agentes basados en utilidad

Figura 113. Tipos de Agentes según su Acción

Otra forma de clasificación es según las acciones de comunicación que realizan para: Intercambiar información, consultar, responder preguntas, recibir peticiones u órdenes, prometer acciones y proponer tratos; aceptar peticiones y propuestas; y, compartir experiencias.

---

[212] Stan Franklin and Art Graesser. *Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents*. Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages, Springer-Verlag, 1996.

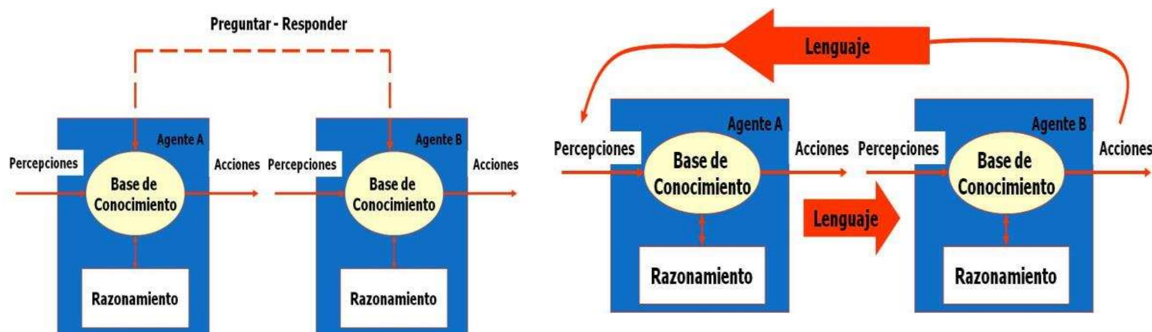


Figura 114. Agentes que se Comunican

Los entornos o ambientes en los que un agente va a desarrollar su acción, pueden ser clasificados como:

- **Accesibles o Inaccesibles.-** Un ambiente se considera accesible si los sensores pueden detectar todos los aspectos que son relevantes para que el agente pueda decidir la acción a tomar.
- **Determinísticos o Estocásticos.-** Un ambiente se dice que es determinístico si el estado siguiente es completamente determinado por el estado actual y las acciones ejecutadas por el agente.
- **Episódicos o No – Episódicos.-** Un ambiente es episódico si las acciones no dependen de las acciones que tuvieron lugar en episodios pasados. Los entornos episódicos son más simples porque el agente no necesita preocuparse de lo que vendrá adelante.
- **Estáticos o Dinámicos.-** Si el ambiente puede cambiar mientras el agente está deliberando acerca de su acción a tomar, entonces se dice que el ambiente es dinámico. Si el ambiente no cambia con el paso del tiempo, pero el desempeño del agente si lo hace, se dice que el ambiente es **semidinámico**.
- **Discretos o Continuos.-** En un ambiente discreto sólo puede existir un limitado número de distintos y claramente definidos preceptos y acciones a seguir. En un ambiente continuo, las variables pueden asumir un rango de valores reales.

Para cada ambiente se requiere de un tipo especial de programa para que el agente se pueda desempeñar favorablemente. Si duda, el caso más complejo es cuando un agente tiene que operar en un ambiente inaccesible, no episódico, dinámico y continuo. En la práctica se ha encontrado que muchas situaciones del ambiente son tan complejas que aunque eventualmente parezca como si fuera verdaderamente determinístico, es preferible tratarlo como estocástico.

12.4 Ingeniería de Agentes Inteligentes

Dentro de las metodologías propuestas para desarrollo de agentes, se pueden distinguir dos enfoques principales:

12.4.1 Extensión de Metodologías Orientadas a Objetos

Incluyen aspectos relacionados con la metodología orientada a objetos e integran elementos relevantes a los agentes.

- **Análisis y Diseño Orientado a Agentes[213]:**
  - Modelo de Agente.- Estructura interna del agente (Creencias, Planes y Objetivos).
  - Modelo Organizacional.- Identificación de roles del agente, elaboración de diagramas utilizando *Object Modeling Technique* (OMT) para jerarquía de herencia y relaciones entre agentes.
  - Modelo de Cooperación.- Tipos de mensajes entre agentes y protocolos utilizados.

- **Técnicas de Modelado de Agentes[214]:**
  - Al nivel externo:
    - Modelo de agentes, para describir la jerarquía de relaciones entre agentes.
    - Modelo de interacción, para describir responsabilidades, servicios e interacciones entre agentes y sistemas externos.
  - Al nivel interno:
    - Modelo de creencias acerca del ambiente
    - Modelo de objetivos y eventos que el agente puede adoptar o responder; y
    - Modelo de planes que el agente puede utilizar para lograr sus objetivos.

- **Método para Multi-Agentes, Basado en Escenarios[215]:**
  - Fase de Análisis:
    - Descripción de escenarios
    - Descripción funcional de los roles
    - Modelado del dato y el mundo
    - Modelado de interacción sistema-usuario.
  - Fase de Diseño:
    - Arquitectura y escenario del sistema
    - Modelado de objetos
    - Modelado de agentes
    - Modelo conversacional y validación del diseño.

---

[213] Burmeister B. *Models and Methodology for Agent-Oriented Analysis and Design*. In K Fisher, editor, Working Notes of the KI'96 Workshop on Agent-Oriented Programming and Distributed Systems, DFKI Document D96-06, 1996.

[214] Kinny D, Georgeff M & Rao A. *A Methodology and Modeling Technique for a System of BDI Agents*. In W Van der Velde & J Perram, editors. Agents Breaking Away: Proceedings of the 7th European Workshop on Modeling Autonomous Agents in a Multi-Agent World. MAAMAW'96. Springer-Verlag, Germany, 1996.

[215] Moulin B & Cloutier L. *Collaborative Work Based on Multi-agent Architecture*. In F Aminzadeh & M Jamshidi, Editors. Soft Computing: Fuzzy Logic, Neural Networks & Distributed Artificial Intelligence. Springer-Verlag, Germany, 1997.

- **Metodología orientada a agentes para integración empresarial[216]:**
  - Modelo funcional
  - Modelo de casos de uso
  - Modelo dinámico
  - El sistema está compuesto de:
    - Identificación de agentes
    - Protocolos de coordinación o scripts.
    - Invocación de planes
    - Creencias, sensores y efectores.

- **Ingeniería de Software Orientada a Agentes[217]:**
  - Análisis del Sistema Orientado a Agentes.- Identificación de agentes que exhiben el comportamiento deseado del sistema. Cada agente tiene sus capacidades que lo caracterizan.
  - Diseño Orientado a Agentes.- Se definen los roles y se integran los agentes identificados en la etapa anterior.
  - Programación Orientada a Objetos.- Se implementa la arquitectura según especificaciones dadas por las etapas anteriores.

12.4.2 Extensión de Metodologías de Ingeniería del Conocimiento

Aprovecha sus técnicas para modelar las características cognitivas de los agentes.

- **Metodología CoMoMAS[218]. Extensión de CommonKADS:**
  - Modelo de Agentes
  - Modelo de Capacidad
  - Modelo de Tarea
  - Modelo de Cooperación
  - Modelo del Sistema
  - Modelo de Diseño

[216] Kendall E A, Malkoun M T & Jiang C. A Methology for Developing Agent Based Systems for Enterprise Integration. In D Luckose & C Zhang, Editors. Proceedings of the First Australian Workshop on DAI, Lecture Notes on Artificial Intelligence, Springer-Verlag, Germany, 1996.

[217] Grosse A. Position Paper. **http://www.AgentLink.org**, DAI Lab, 1995.

[218] Glaser N. *Contribution to Knowledge Modeling in a Multi-Agent Framework (The CoMoMAS Approach)*. PhD Thesis, L'Université Henri Poicaré, France, Nov, 1996.

- **Metodología MAS-CommonKADS**[219]**:**
  - Modelo de agentes
  - Modelo de tarea
  - Modelo de capacidades
  - Modelo de coordinación
  - Modelo de organización
  - Modelo de comunicación
  - Modelo de diseño
  - Diseño de la plataforma.

- **Top-down Object-based Goal-oriented Approach (TOGA)**[220]

  Es una herramienta de organización del conocimiento orientado a metas (modelación conceptual) para la especificación e identificación de procesos o sistemas de problemas complejos del mundo real. En este sentido, puede ser visto como un meta-modelo inicial, genérico, basado en axioma y, posteriormente, como una metodología de descomposición de problemas y especialización, utilizando conocimiento disponible.

  Para el diseñador de sistemas de ingeniería compleja, **TOGA** provee una conceptualización basada en agentes inteligentes con un conjunto estructurado de métodos y reglas que le permiten controlar las actividades y procesos de modelación conceptual descendente y orientada a metas. Esto posibilita la especificación formal e implementación de sistemas basados en agentes.

12.5 Aplicaciones de los Agentes Inteligentes

Los sistemas computarizados autónomos de ninguna manera son desarrollos nuevos. Ejemplos de tales sistemas son:

- Cualquier sistema digital de control de procesos, que deben monitorear un entorno real y ejecutar acciones en tiempo real para modificar ciertas variables, a medida que las condiciones cambian.
- Los *daemons* que monitorean un entorno de software y ejecutan acciones para modificarlo de acuerdo con las condiciones que hayan cambiado.

Pero estas aplicaciones si bien se las denomina agentes, no son inteligentes. Para que un agente sea considerado inteligente, debe tener flexibilidad para realizar acciones autónomas que le permitan alcanzar los objetivos para los que fue diseñado. En este contexto, se entiende que para que sea flexible, el sistema debe ser:

---

[219] Iglesias C A, Garijo M, González J C & Velasco J R. *Analysis & Design of Multiagent Systems using MAS-CommonKADS*. In AAAI'97 Workshop on Agent Theories, Architectures & Languages, Providence, July 1997.

[220] http://erg4146.casaccia.enea.it/wwwerg26701/Gad-toga.htm

- Capaz de percibir su entorno (que puede ser el mundo físico, un usuario, otros agentes, el Internet, etc.) y responder ágilmente a los cambios que ocurran.
- Proactivo, es decir no debe simplemente actuar en respuesta a los cambios en su entorno, sino que debe exhibir un comportamiento de acuerdo a la oportunidad, guiado por metas y tomar la iniciativa cuando sea apropiado; y,
- Social, para que interactúe, cuando considere apropiado, con otros agentes artificiales y humanos para resolver sus propios problemas y apoyar a otros en sus iniciativas.

Existen diversas dimensiones ortogonales que pueden ser utilizadas para clasificar a los agentes inteligentes: por el tipo de agente, por la tecnología utilizada para su implementación o por el dominio de su aplicación. En este caso se utilizará esta última opción.

12.5.1 Aplicaciones Industriales

Las aplicaciones industriales de las tecnologías de agentes están entre las que primero se desarrollaron:

- **Control de Procesos.-** Al ser los controladores de procesos sistemas autónomos reactivos, constituyen una aplicación natural de los agentes inteligentes y sistemas multi-agente. Muchas aplicaciones se han desarrollado, incluyendo una plataforma de software para el desarrollo de sistemas multiagente. *ARchitecture for Cooperative Heterogeneous ON-Line systems* (**ARCHON**)[221] es un marco de trabajo de software que aporta con herramientas y una metodología para la estructuración y el diseño de las interacciones entre los subcomponentes de un sistema distribuido de inteligencia artificial. Algunas de las aplicaciones exitosas incluyen: sistemas de generación, transmisión y distribución de electricidad; control de una planta de cemento; control de un acelerador de partículas y control de aplicaciones robóticas. El tipo de cooperación comunitaria que soporta tiene un régimen de control descentralizado y agentes individuales para la solución de problemas. Estos agentes están débilmente acoplados y son semi-autónomos. La plataforma ARCHON tiene 4 grupos de herramientas: para crear instancias de agentes; para crear instancias de los meta-datos; para trabajar en línea; y, para explorar y depurar datos[222].
- **Manufactura.- YAMS** (*Yet Another Manufacturing System*), desarrollado por Van Dyke Parunak[223], utiliza el Contract Net Interaction Protocol[224] para el control de sistemas de manufactura. Una empresa manufacturera puede ser modelada como una jerarquía de celdas de trabajo. YAMS adopta los multi-agentes donde cada fábrica y sus componentes (celdas de trabajo) son representados como un agente. La meta de YAMS

---

[221] http://eprints.ecs.soton.ac.uk/2105/1/IEEE-Expert-part1.pdf
[222] http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.36.8941&rep=rep1&type=pdf
[223] http://www.springerlink.com/index/262p467068782563.pdf
[224] http://www.ensc.sfu.ca/idea/courses/files/Contract%20Net%20Protocol1.pdf

es gestionar eficientemente los procesos de producción en estas plantas. Un sistema multi-agente de control de fabricación usualmente requiere del uso de técnicas especiales de razonamiento y coordinación. Dependiendo de los objetivos de la producción y los tipos de procesos de producción, pueden ser necesarias diferentes arquitecturas de control y estrategias a fin de controlar de manera óptima el proceso de fabricación. Sin embargo, a pesar de las necesidades especiales de una aplicación de fabricación particular, cualquier sistema de control industrial debe satisfacer unos requisitos generales[225].

- **Control de Tráfico Aéreo.-** Un sistema para control de tráfico aéreo basado en agentes denominado **OASIS**[226], se aplicó en el aeropuerto de Sidney, Australia. El sistema permite la implementación de agentes utilizando el modelo *belief-desire-intention* (***BDI***)[227]. Agentes son utilizados para representar tanto aeroplanos como sistemas de control de tráfico aéreo que están en operación.

12.5.2 Aplicaciones Comerciales

- **Gestión de la Información.-** En esta época caracterizada por la sobrecarga de información, los agentes han encontrado el espacio propicio para apoyar a los usuarios en la recopilación y filtraje de la información disponible en la WEB. **MAXIMS** es un agente que filtra el correo electrónico y **Newt** es otro agente que aprende a seleccionar noticias que un usuario debe o no leer, a base de ejemplos[228]. **Zuno Digital Library**, es un sistema multi-agente que ayuda a un usuario a encontrar información relevante, blindándolo de información que no es de su interés[229].
- **Comercio Electrónico.**- Poco a poco se va llegando a situaciones en las ciertos procesos de negociación son encargados a agentes de promoción, mercadeo, compra y venta. Un mercado electrónico denominado **Kasbah[230]**, es un ejemplo en el que agentes de compra y venta son los que realizan las transacciones comerciales de cada artículo disponible.
- **Gestión de Procesos de Negocios (BPM).**- Obtener información pertinente y actualizada para que los gerentes puedan tomar decisiones en una corporación, puede ser un proceso complejo y lento. Por esta razón las organizaciones han desarrollado diferentes sistemas de información para que les asistan en los diferentes aspectos de la gestión empresarial. El proyecto **ADEPT**[231] trata de resolver estos problemas visualizando los procesos del negocio como una comunidad de agentes negociadores y

[225] **http://www.fi.uba.ar/materias/7565/U5-industria.pdf**
[226] Kinny, D.; Ljungberg, M.; Rao, A-; Sonenberg. L; Tidhard, G.; and Werner, E.. *Planned Team Activity*. In Artificial Social Systems, eds. C. Castelfranchi and E. Werner. New York: Springer-Verlag, 1992.
[227] **http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.41.3852&rep=rep1&type=pdf**
[228] Maes, P. *Agents that reduce work and information overload*. Communications of the ACM, 37(7), 1994, pp. 31-40.
[229] Zuno Ltd (1997) Ver **http://www.dlib.com**
[230] Chavez, A., Maes, P. *Kasbah: An agent marketplace for buying and selling goods*. Proceedings of First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Systems, London, UK, 1996.
[231] Jennings, N. R., Faratin, P., Johnson, M. J., Norman, T. J., O'Brien, P., Wiegand, M. E. *Agent-based business process management*. Int. Journal of Cooperative Information Systems, 5(2, 3), 1996, pp. 105-130.

proveedores de servicio. Cada agente representa un rol distinto o un departamento en la empresa y es capaz de proveer uno o más servicios. Esta solución basada en agentes, ofrece numerosas ventajas sobre las soluciones típicas a estos problemas. El sistema se ha probado en procesos de negocio en British Telecom involucrando 200 actividades y nueve departamentos.

12.5.3 Aplicaciones Médicas

La informática médica es una de las áreas de mayor crecimiento en las ciencias de computación. Dos de las primeras aplicaciones en este campo están las áreas de cuidados de salud y monitoreo de pacientes.

- **Monitoreo de Pacientes.-** El sistema **Guardian**[232] fue desarrollado para gestionar el cuidado de pacientes en la unidad quirúrgica de cuidados intensivos (SICU). El sistema distribuye la función de monitoreo de pacientes entre un número de agentes de tres tipos diferentes: agentes percepción/acción; agentes de razonamiento; y, agentes de control. Los agentes son organizados jerárquicamente y el sistema está basado en un modelo de control tipo pizarra[233].
- **Cuidados de Salud.**- Un prototipo de sistema distribuido basado en agentes ha sido desarrollado por **Huang et al**[234]**.** El prototipo tiene un sistema basado en conocimiento en donde está contenida la experticia de los agentes; una interfaz humano – computador a través de la cual el usuario agrega, remueve o revisa las metas del sistema; y, una unidad que administra las comunicaciones entre agentes. El componente inteligente del sistema está basado en el modelo **KADS** (*Knowledge Analysis and Design System*) y la arquitectura del sistema está implementada en PROLOG.

12.5.4 Entretenimiento

La industria del entretenimiento a menudo no es tomada muy en serio por la comunidad científica en computación. Estas aplicaciones son vistas como algo periférico a las aplicaciones serias. Sin embargo, este es un campo sumamente lucrativo. Además, existen múltiples oportunidades para la aplicación de agentes inteligentes.

- **Juegos.- Wavish et al.**[235] Describen varias aplicaciones de la tecnología de agentes a los juegos de computadora. Ellos utilizan un modelo de agente reactivo llamado ***RTA*** (***Real Time Able***), en esta aplicación los agentes están programados en términos de

---

[232] Hayes-Roth, B., Hewett, M., Washington, R., Hewett, R., Seiver, A. *Distributing intelligence within an individual*. In: L. Gasser, M. N. Huhns (Eds.) Distributed AI, Volume II, 385-412. Morgan Kaufmann, 1995.

[233] **http://intelec.dif.um.es/~aike/roque/pub/eis98-control.pdf**

[234] Huang, J., Jennings, N. R., Fox, J. *An agent-based approach to health care management*. Int. Journal of Applied Artificial Intelligence, 9(4), 1995, pp. 401-420.

[235] Wavish, P., Graham, M. A situated action approach to implementing characters in computer games. Int. Journal of Applied Artificial Intelligence, 10(1), 1996, pp. 53-74.

comportamientos. Estos son representados como simples estructuras que reflejan reglas pero no requieren razonamiento simbólico complejo.

- **Teatro y Cine Interactivo.**[236] Estas aplicaciones implican un sistema que permiten que un usuario pueda interpretar un papel similar al que actores reales lo hacen en obras de cine o teatro, interactuando con caracteres artificiales generados por la computadora, pero que tiene comportamientos similares a los de la gente real. Diversos proyectos se están ejecutando para investigar el desarrollo de tales agentes.

12.6 Conclusión

El campo de la inteligencia artificial en general, y el de los agentes, en particular están en una dinámica evolución. El diseño de sistemas robustos y eficientes es una de las tareas más importantes que preocupa a los ingenieros de software, y un enorme esfuerzo se ha desplegado para hacer del diseño de sistemas un proceso metódico y riguroso. Todavía pocas personas en la comunidad de investigadores de agentes han considerado cómo se puede aplicar metodologías de diseño a los sistemas basados en agentes. Claramente, esta situación tiene que cambiar si se desea que los sistemas basados en agentes sean ampliamente aceptados y sea posible explotar todo el potencial que prometen estas aplicaciones.

[236] http://www.lite.etsii.urjc.es/liliana/agentes.pdf

13 SITIOS WEB DE INTERÉS

- **Wikipedia – The Free Encyclopedia**
    - **www.wikipedia.org**
- **Institute of Electrical & Electronis Engineers - USA (IEEE, www.ieee.org):**
    - **Computer Society: www.computer.org**
    - **Computational Intelligence Society: ieee-cis.org**
- **Asociation for Computer Machinery – USA:**
    - **www.acm.org**
    - **www.acm.org/crossroads**
- **Inteligencia Artificial y Robótica**

  http://www.inteligenciaartificial.cl/
- **Asociación Española de Inteligencia Artificial.**

  http://www.aepia.org/
- **Revista Iberoamericana de inteligencia artificial.**

  http://cabrillo.lsi.uned.es:8080/aepia
- **Informática Industrial e Inteligencia Artificial.**

  http://www.i3a.ua.es/
- **European Co-ordination Action for Agent Based Computing**

  http://www.AgentLink.org

14 BIBLIOGRAFÍA

- **M. Tim Jones.** *Artificial Intelligence: A Systems Approach with CD*. **Jones and Bartlett Publishers, Inc; 1st edition; December 26, 2008.**

- **Stuart Russell, Peter Norvig.** *Artificial Intelligence: A Modern Approach*. **Prentice Hall; 2nd edition; December 30, 2002.**

- **Sue Ellen Haupt, Antonello Pasini, and Caren Marzban.** *Artificial Intelligence Methods in the Environmental Sciences*. **Springer; 1st edition, December 12, 2008.**

- **Dario Floreano, Claudio Mattiussi.** *Bio-Inspired Artificial Intelligence: Theories, Methods, and Technologies (Intelligent Robotics and Autonomous Agents)*. **The MIT Press; September 30, 2008**

- **Blay Whitby.** *Artificial Intelligence: A Beginner's Guide*. **Oneworld Publications; September 25, 2008**

- **George F. Luger, William A Stubblefield.** *AI Algorithms, Data Structures, and Idioms in Prolog, Lisp, and Java for Artificial Intelligence: Structures and Strategies for Complex Problem Solving*. **Addison Wesley; 6 edition; September 4, 2008.**

- **George F. Luger.** *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*. **Addison Wesley; 6th edition; March 7, 2008.**

- **Marvin Minsky.** *The Emotion Machine: Commonsense Thinking, Artificial Intelligence, and the Future of the Human Mind*. **Simon & Schuster; November 13, 2007.**

- **M. Tim Jones.** *AI Application Programming*. **Charles River Media; 2nd edition; June 3, 2005**

- **Leandro Nunes de Castro.** *Fundamentals of Natural Computing: Basic Concepts, Algorithms, and Applications*. **Chapman & Hall/CRC; 1st edition; June 2, 2006.**

15 GLOSSARY[237]

A

Artificial Intelligence

Artificial intelligence (AI) is the mimicking of human thought and cognitive processes to solve complex problems automatically. AI uses techniques for writing computer code to represent and manipulate knowledge. Different techniques mimic the different ways that people think and reason (see **case-based reasoning** and **model-based reasoning** for example). AI applications can be either stand-alone software, such as decision support software, or embedded within larger software or hardware systems.

AI has been around for about 50 years and while early optimism about matching human reasoning capabilities quickly has not been realized yet, there is a significant and growing set of valuable applications. AI has not yet mimicked much of the common-sense reasoning of a five-year old child. Nevertheless, it can successfully mimic many expert tasks performed by trained adults, and there is probably more artificial intelligence being used in practice in one form or another than most people realize.

Intelligent applications will only be achievable with artificial intelligence and it is the mark of a successful designer of AI software to deliver functionality that cannot be delivered without using AI.

Action-based Planning

The goal of action-based planning is to determine how to decompose a high-level action into a network of subactions that perform the requisite task. Therefore the major task within such a planning system is to manage the constraints that apply to the interrelationships (e.g., ordering constraints) between actions. In fact, action-based planning is best viewed as a **constraint satisfaction** problem.

The search for a plan cycles through the following steps: choose a constraint and apply the constraint check; if the constraint is not satisfied, choose a bug from the set of constraint bugs; choose and apply a fix, yielding a new plan and possibly a new set of constraints to check.

In contrast, state-based planners generally conduct their search for a plan by reasoning about how the actions within a plan affect the state of the world and how the state of the world affects the applicability of actions.

Adaptive Interface

A computer interface that automatically and dynamically adapts to the needs and competence of each individual user of the software.

Agents

Agents are software programs that are capable of autonomous, flexible, purposeful and reasoning action in pursuit of one or more goals. They are designed to take timely action in response to external stimuli from their environment on behalf of a human. When multiple agents are being used together in a system, individual agents are expected to interact together as appropriate to achieve the goals of the overall system. Also called autonomous agents, assistants, brokers, bots, droids, intelligent agents, software agents.

Agent Architecture

There are two levels of agent architecture, when a number of agents are to work together for a common goal. There is the architecture of the system of agents, that will determine how they work together, and which does not need to be concerned with how individual agents fulfil their sub-missions; and the architecture of each individual agent, which does determine its inner workings.

---

[237] http://www.stottlerhenke.com/ai_general/glossary.htm

The architecture of one software agent will permit interactions, among most of the following components (depending on the agent's goals): perceptors, effectors, communication channels, a state model, a model-based reasoner, a planner/scheduler, a reactive execution monitor, its reflexes (which enable the agent to react immediately to changes in its environment that it can't wait on the planner to deal with), and its goals. The perceptors, effectors, and communication channels will also enable interaction with the agent's outside world.

AI Effect

The great practical benefits of AI applications and even the existence of AI in many software products go largely unnoticed by many despite the already widespread use of AI techniques in software. This is the AI effect. Many marketing people do not use the term "artificial intelligence" even when their company's products rely on some AI techniques. Why not? It may be because AI was oversold in the first giddy days of practical rule-based expert systems in the 1980s, with the peak perhaps marked by the Business Week cover of July 9, 1984 announcing, Artificial Intelligence, IT'S HERE.

James Hogan in his book, Mind Matters, has his own explanation of the AI Effect:

"AI researchers talk about a peculiar phenomenon known as the "AI effect." At the outset of a project, the goal is to entice a performance from machines in some designated area that everyone agrees would require "intelligence" if done by a human. If the project fails, it becomes a target of derision to be pointed at by the skeptics as an example of the absurdity of the idea that AI could be possible. If it succeeds, with the process demystified and its inner workings laid bare as lines of prosaic computer code, the subject is dismissed as "not really all that intelligent after all." Perhaps ... the real threat that we resist is the further demystification of ourselves...It seems to happen repeatedly that a line of AI work ... finds itself being diverted in such a direction that ... the measures that were supposed to mark its attainment are demonstrated brilliantly. Then, the resulting new knowledge typically stimulates demands for application of it and a burgeoning industry, market, and additional facet to our way of life comes into being, which within a decade we take for granted; but by then, of course, it isn't AI."

AI Languages and Tools

AI software has different requirements from other, conventional software. Therefore, specific languages for AI software have been developed. These include LISP, Prolog, and Smalltalk. While these languages often reduce the time to develop an artificial intelligence application, they can lengthen the time to execute the application. Therefore, much AI software is now written in languages such as C++ and Java, which typically increases development time, but shortens execution time. Also, to reduce the cost of AI software, a range of commercial software development tools have also been developed. Stottler Henke has developed its own proprietary tools for some of the specialized applications it is experienced in creating.

Algorithm
An algorithm is a set of instructions that explain how to solve a problem. It is usually first stated in English and arithmetic, and from this, a programmer can translate it into executable code (that is, code to be run on a computer).

Applications of Artificial Intelligence

The actual and potential applications are virtually endless. Reviewing Stottler Henke's work will give you some idea of the range. In general, AI applications are used to increase the productivity of knowledge workers by intelligently automating their tasks; or to make technical products of all kinds easier to use for both workers and consumers by intelligent automation of different aspects of the functionality of complex products.

Associative Memories

Associative memories work by recalling information in response to an information cue. Associative memories can be autoassociative or heteroassociative. Autoassociative memories recall the same information that is used as a cue, which can be useful to complete a partial pattern. Heteroassociative memories are useful as a memory. Human long-term memory is thought to be associative because of the way in which one thought retrieved from it leads to another. When we want to store a new item of information in our long term memory it

typically takes us 8 seconds to store an item that can't be associated with a pre-stored item, but only one or two seconds, if there is an existed information structure with which to associate the new item.

Automated Diagnosis Systems

Most diagnosis work is done by expert humans such as mechanics, engineers, doctors, firemen, customer service agents, and analysts of various kinds. All of us usually do at least a little diagnosis even if it isn't a major part of our working lives. We use a range of techniques for our diagnoses. Primarily, we compare a current situation with past ones, and reapply, perhaps with small modifications, the best past solutions. If this doesn't work, we may run small mental simulations of possible solutions through our minds, based on first principles. We may do more complex simulations using first principles on paper or computers looking for solutions. Some problems are also amenable to quantitative solutions. We may hand off the problem to greater experts than ourselves, who use the same methods. The problem with humans doing diagnosis is that it often takes a long time and a lot of mistakes to learn to become an expert. Many situations just don't reoccur frequently, and we may have to encounter each situation several time to become familiar with it. Automatic diagnosis systems can help avoid these problems, while helping humans to become experts faster. They work best in combination with a few human experts, as there are some diagnosis problems that humans are better at solving, and also because humans are more creative and adaptive than computers in coming up with new solutions to new problems.

Autonomous Agents

A piece of AI software that automatically performs a task on a human's behalf, or even on the behalf of another piece of AI software, so together they accomplish a useful task for a person somewhere. They are capable of independent action in dynamic, unpredictable environments. "Autonomous agent" is a trendy term that is sometimes reserved for AI software used in conjunction with the Internet (for example, AI software that acts as your assistance in intelligently managing your e-mail).

Autonomous agents present the best hope from gaining additional utility from computing facilities. Over the past few years the term "agent" has been used very loosely. Our definition of a software agent is: "an intelligent software application with the authorization and capability to sense its environment and work in a goal directed manner." Generally, the term "agent" implies "intelligence", meaning the level of complexity of the tasks involved approaches that which would previously have required human intervention.

B

Bayesian Networks

A modeling technique that provides a mathematically sound formalism for representing and reasoning about uncertainty, imprecision, or unpredictability in our knowledge. For example, seeing that the front lawn is wet, one might wish to determine whether it rained during the previous night. Inference algorithms can use the structure of the Bayesian network to calculate conditional probabilities based on whatever data has been observed (e.g., the street does not appear wet, so it is 90% likely that the wetness is due to the sprinklers). Bayesian networks offer or enable a set of benefits not provided by any other system for dealing with uncertainty - an easy to understand graphical representation, a strong mathematical foundation, and effective automated tuning mechanisms. These techniques have proved useful in a wide variety of tasks including medical diagnosis, natural language understanding, plan recognition, and intrusion detection. Also called belief networks, Bayes networks, or causal probabalistic networks.

C

Case-based Reasoning

Case-based reasoning (CBR) solves a current problem by retrieving the solution to previous similar problems and altering those solutions to meet the current needs. It is based upon previous experiences and patterns of previous experiences. Humans with years of experience in a particular job and activity (e.g., a skilled paramedic

arriving on an accident scene can often automatically know the best procedure to deal with a patient) use this technique to solve many of their problems. One advantage of CBR is that inexperienced people can draw on the knowledge of experienced colleagues, including ones who are not in the organization, to solve their problems. Synonym: Reasoning by analogy.

Classification

Automated classification tools such as **decision trees** have been shown to be very effective for distinguishing and characterizing very large volumes of data. They assign items to one of a set of predefined classes of objects based on a set of observed features. For example, one might determine whether a particular mushroom is "poisonous" or "edible" based on its color, size, and gill size. Classifiers can be learned automatically from a set of examples through **supervised learning**. Classification rules are rules that discriminate between different partitions of a database based on various attributes within the database. The partitions of the database are based on an attribute called the classification label (e.g., "faulty" and "good").

Clustering

Clustering is an approach to learning that seeks to place objects into meaningful groups automatically based on their similarity. Clustering, unlike **classification**, does not require the groups to be predefined with the hope that the algorithm will determine useful but hidden groupings of data points. The hope in applying clustering algorithms is that they will discover useful but unknown classes of items. A well-publicized success of a clustering system was NASA's discovery of a new class of stellar spectra. See IQE, GIIF, WebMediator, Rome Graphics, and data mining for examples of applications that use clustering.

Cognitive Science

Artificial intelligence can be defined as the mimicking of human thought to perform useful tasks, such as solving complex problems. This creation of new paradigms, algorithms, and techniques requires continued involvement in the human mind, the inspiration of AI. To that end, AI software designers team with cognitive psychologists and use cognitive science concepts, especially in knowledge elicitation and system design.

Cognitive Task Analysis

Cognitive task analysis (CTA) is a systematic process by which the cognitive elements of task performance are identified. This includes both domain knowledge and cognitive processing. Thus, CTA focuses on mental activities that cannot be observed and is in contrast to behavioral task analysis that breaks the task down into observable, procedural steps. CTA is most useful for highly complex tasks with few observable behaviors. Examples of cognitive processing elements include: to decide, judge, notice, assess, recognize, interpret, prioritize, and anticipate. Examples of domain knowledge elements include concepts, principles, and interrelationships; goals and goal structures; rules, strategies and plans; implicit knowledge; and mental models.

The results from CTA have various applications such as identifying content to be included within training programs for complex cognitive tasks, research on expert-novice differences in terms of domain knowledge and cognitive processing during task performance, modeling of expert performance to support expert system design, and the design of human-machine interfaces.

Collaborative Filtering

A technique for leveraging historical data about preferences of a body of users to help make recommendations or filter information for a particular user. Intuitively, the goal of these techniques is to develop an understanding of what may be interesting to a user by uncovering what is interesting to people who are similar to that user. See GIIF and IQE for examples of applications that use collaborative filtering techniques.

Commonsense Reasoning

Ordinary people manage to accomplish an extraordinary number of complex tasks just using simple, informal thought processes based on a large amount of common knowledge. They can quickly plan and undertake a shopping expedition to six or seven different shops, as well as pick up the kids from soccer and drop a book back at the library, quite efficiently without logically considering the hundreds of thousands of alternative ways to plan such an outing. They can manage their personal finances, or find their way across a crowded room dancing without hitting anyone, just using commonsense reasoning. Artificial intelligence is far behind humans

in using such reasoning except for limited jobs, and tasks that rely heavily on commonsense reasoning are usually poor candidates for AI applications.

Computer Vision

Making sense of what we see is usually easy for humans, but very hard for computers. Practical vision systems to date are limited to working in tightly controlled environments. Synonym: machine vision

Constraint Satisfaction

Constraints are events, conditions or rules that limit our alternatives for completing a task. For example, the foundation of a building has to be laid before the framing is done; a car has to be refueled once every four hundred miles, a neurosurgeon is needed to perform brain surgery, or a Walkman can only operate on a 9-volt battery.

Satisfying constraints is particularly important in scheduling complex activities. By first considering applicable constraints, the number of possible schedules to be considered in a search for an acceptable schedule can be reduced enormously, making the search process much more efficient. Constraint satisfaction techniques can be used to solve scheduling problems directly. Constraint satisfaction algorithms include heuristic constraint-based search and annealing.

D

Data Fusion

Information processing that deals with the association, correlation, and combination of data and information from single and multiple sources to achieve a more complete and more accurate assessment of a situation. The process is characterized by continuous refinement of its estimates and assessments, and by evaluation of the need for additional sources, or modification of the process itself, to achieve improved results.

Data Mining

The non-trivial process of uncovering interesting and useful relationships and patterns in very large databases to guide better business and technical decisions. Data mining is becoming increasingly important due to the fact that all types of commercial and government institutions are now logging huge volumes of data and now require the means to optimize the use of these vast resources. The size of the databases to which data mining techniques are applied is what distinguishes them from more traditional statistical and machine learning approaches, which can be computationally costly. Data mining forms part of the overall process of 'Knowledge Discovery in Databases.' Data mining is preceded by the preliminary stages of preparing and cleaning up the data, and followed by the subsequent incorporation of other relevant knowledge, and the final interpretation. See all the data mining projects for examples of use of data mining techniques.

Decision Aids

Software that helps humans make decisions, particularly about complex matters when a high degree of expertise is needed to make a good decision.

Decision Support

Decision support is a broad class of applications for artificial intelligence software. There are many situations when humans would prefer machines, particularly computers, to either automatically assist them in making decisions, or actually make and act on a decision. There are a wide range of non-AI decision support systems such as most of the process control systems successfully running chemical plants and power plants and the like under steady state conditions. However, whenever situations become more complex—for example, in chemical plants that don't run under steady state, or in businesses when both humans and equipment are interacting—intelligent decision support is required. That can only be provided by automatic decision support software using artificial intelligence techniques. Stottler Henke has created a wide range of decision support applications that provide examples of such situations. Synonym: intelligent decision support.

Decision Theory

Decision theory provides a basis for making choices in the face of uncertainty, based on the assignment of probabilities and payoffs to all possible outcomes of each decision. The space of possible actions and states of the world is represented by a decision tree.

Decision Trees

A decision tree is a graphical representation of a hierarchical set of rules that describe how one might evaluate or classify an object of interest based on the answers to a series of questions. For instance, a decision tree can codify the sequence of tests a doctor might take in diagnosing a patient. Such a decision tree will order the tests based on their importance to the diagnostic task. The result of each successive test dictates the path you take through the tree and therefore the tests (and their order) that will be suggested. When you finally reach a node in which there is no further tests are suggested, the patient has been fully diagnosed. Decision trees have the advantage of being easy to understand because of their hierarchical rule structure, and explanations for their diagnoses can be readily and automatically generated.

Decision trees can be automatically developed from a set of examples and are capable of discovering powerful predictive rules even when very large numbers of variables are involved. These algorithms operate by selecting the test that best discriminates amongst classes/diagnoses and then repeating this test selection process on each of the subsets matching the different test outcomes (e.g., "patients with temperatures greater than 101ºF" and "patients with temperatures less than or equal to 101ºF"). This process continues until all the examples in a particular set have the same class/diagnosis.

Dependency Maintenance

Dependency maintenance is the technique of recording why certain beliefs are held, decisions were made, or actions were taken, in order to facilitate revising those decisions, actions, or beliefs in the face of changing circumstances. Several families of **truth maintenance systems** have been developed to facilitate dependency maintenance in particular kinds of situations (e.g. need to consider many alternate scenarios versus a single scenario, frequency with which assumptions change, etc).

Document Clustering

With document clustering techniques, documents can be automatically grouped into meaningful classes so that users of a database of full-text documents can easily search through related documents.Finding individual documents from amongst large on-line, full-text collections has been a growing problem in recent years due to the falling price of computer storage capacity and the networking of document databases to large numbers of people. Traditional library indexing has not provided adequate information retrieval from these large sources. The techniques for document clustering generally involve some natural language processing along with a collection of statistical measures.

Domain

An overworked word for AI people. "Domain" can mean a variety of things including a subject area, field of knowledge, an industry, a specific job, an area of activity, a sphere of influence, or a range of interest, e.g., chemistry, medical diagnosis, putting out fires, operating a nuclear power plant, planning a wedding, diagnosing faults in a car. Generally, a domain is a system in which a particular set of rules, facts, or assumptions operates. Humans can usually easily figure out what is meant from the context in which "domain" is used; computers could probably not figure out what a human means when he or she says "domain."

Domain Expert

The person who knows how to perform an activity within the domain, and whose knowledge is to be the subject of an expert system. This person's or persons' knowledge and method of work are observed, recorded, and entered into a knowledge base for use by an expert system. The domain expert's knowledge may be supplemented by written knowledge contained in operating manuals, standards, specifications, computer programs, etc., that are used by the experts. Synonym: subject-matter expert (SME).

E

Emergence
Emergence is the phenomenon of complex patterns of behavior arising out of the myriad interactions of simple agents, which may each operate according to a few simple rules. To put it another way, an emergent system is much more than simply the sum of its parts. It can happen without any grand master outside the system telling the individual agents how to behave. For example, all the people in a modern city acting in their individual capacities as growers, processors, distributors, sellers, buyers, and consumers of food collectively create a food market matching supply and demand of thousands of different items, without an overall plan. An ant colony provides another example of simple agents, each operating according to a few simple rules, producing a larger system that finds food, provide shelter and protection for its members. Artificial intelligence software running on powerful computers can demonstrate useful emergent behavior as well, such as that demonstrated in automatic scheduling software that creates near-optimal schedules for complex activities subject to many constraints.

Expert System

An expert system encapsulates the specialist knowledge gained from a human expert (such as a bond trader or a loan underwriter) and applies that knowledge automatically to make decisions. For example, the knowledge of doctors about how to diagnose a disease can be encapsulated in software. The process of acquiring the knowledge from the experts and their documentation and successfully incorporating it in the software is called knowledge engineering, and requires considerable skill to perform successfully. Applications include customer service and helpdesk support, computer or network troubleshooting, regulatory tracking, autocorrect features in word processors, document generation such as tax forms, and scheduling.

F

Fuzzy Logic

Traditional Western logic systems assume that things are either in one category or another. Yet in everyday life, we know this is often not precisely so. People aren't just short or tall, they can be fairly short or fairly tall, and besides we differ in our opinions of what height actually corresponds to tall, anyway.. The ingredients of a cake aren't just not mixed or mixed, they can be moderately well mixed. Fuzzy logic provides a way of taking our commonsense knowledge that most things are a matter of degree into account when a computer is automatically making a decision. For example, one rice cooker uses fuzzy logic to cook rice perfectly even if the cook put in too little water or too much water.

G

Game Theory

Game theory is a branch of mathematics that seeks to model decision making in conflict situations.

Genetic Algorithms

Search algorithms used in machine learning which involve iteratively generating new candidate solutions by combining two high scoring earlier (or parent) solutions in a search for a better solution. So named because of its reliance on ideas drawn from biological evolution.

Granularity
Refers to the basic size of units that can be manipulated. Often refers to the level of detail or abstraction at which a particular problem is analyzed. One characteristic of human intelligence, Jerry R. Hobbs has pointed out, is the ability to conceptualize a world at different levels of granularity (complexity) and to move among them

in considering problems and situations. The simpler the problem, the coarser the grain can be and still provide effective solutions to the problem.

H

Heterogeneous Databases

Databases that contain different kinds of data, e.g, text and numerical data.

Heuristic

A heuristic is commonly called a rule of thumb. That is, a heuristic is a method for solving a problem that doesn't guarantee a good solution all the time, but usually does. The term is attributed to the mathematician, George Polya. An example of a heuristic would be to search for a lost object by starting in the last place you can remember using it.

Human-Centered Computing

Computers and other machines should be designed to effectively serve people's needs and requirements. All too often they're not. Commonly cited examples of this are the difficulty people have in setting up their VCR to record a TV show; and the difficulties people have in setting up a home computer facility, or hooking up to the Internet. Artificial intelligence software can be used to deliver more human-centered computing, improving system usability, extending the powerfulness of human reasoning and enabling greater collaboration amongst humans and machines, and promoting human learning. A goal of human-centered computing is for cooperating humans and machines to compensate for each other's respective weaknesses (e.g., machines to compensate for limited human short-term memory and the slowness with which humans can search through many alternative possible solutions to a problems; and for humans to compensate machines for their more limited pattern-recognition capability, language processing, and creativity) in support of human goals.

Synonym: mixed initiative planning.

Hybrid Systems

Many of Stottler Henke's artificial intelligence software applications use multiple AI techniques in combination. For example, case-based reasoning may be used in combination with model-based reasoning in an automatic diagnostic system. Case-based reasoning, which tends to be less expensive to develop and faster to run, may draw on an historical databases of past equipment failures, the diagnosis of those, and the repairs effected and the outcomes achieved. So CBR may be used to make most failure diagnoses. Model-based reasoning may be used to diagnose less common but expensive failures and to make fine adjustments to the repair procedures retrieved from similar cases in the case base by CBR.

Inference Engine

The part of an expert system responsible for drawing new conclusions from the current data and rules. The inference engine is a portion of the reusable part of an expert system (along with the user interface, a knowledge base editor, and an explanation system), that will work with different sets of case-specific data and knowledge bases.

Information Filtering

An information filtering system sorts through large volumes of dynamically generated information to present to the user those nuggets of information, which are likely to satisfy his or her immediate needs. Information filtering overlaps the older field of information retrieval, which also deals with the selection of information. Many of the features of information retrieval system design (e.g. representation, similarity measures or Boolean selection, document space visualization) are present in information filtering systems as well. Information filtering is roughly information retrieval from a rapidly changing information space.

Intelligent Tutoring Systems

Encode and apply the subject matter and teaching expertise of experienced instructors, using artificial intelligence (AI) software technologies and cognitive psychology models, to provide the benefits of one-on-one instruction -- automatically and cost-effectively. These systems provide coaching and hinting, evaluate each student's performance, assess the student's knowledge and skills, provide instructional feedback, and select appropriate next exercises for the student.

K

KAPPA
Rule-based object-oriented expert system tool and application developer (IntelliCorp Inc.). KAPPA is written in C, and is available for PCs. See AI Languages and Tools.

Knowledge-based Representations

The form or structure of databases and knowledge bases for expert and other intelligent systems, so that the information and solutions provided by a system are both accurate and complete. Usually involves a logically based language capable of both syntactic and semantic representation of time, events, actions, processes, and entities. Knowledge representation languages include Lisp, Prolog, Smalltalk, OPS-5, and KL-ONE. Structures include rules, scripts, frames, endorsements, and semantic networks.

Knowledge-based Systems

Usually a synonym for expert system, though some think of expert systems as knowledge-based systems that are designed to work on practical, real-world problems.

Knowledge Elicitation

Synonym: knowledge acquisition.

Knowledge Engineering

Knowledge engineering is the process of collecting knowledge from human experts in a form suitable for designing and implementing an expert system. The person conducting knowledge engineering is called a knowledge engineer.

Knowledge Fusion

See **Data Fusion**.

Knowledge Representation

Knowledge representation is one of the two basic techniques of artificial intelligence; the other is the capability to search for ends from a starting point. The way in which knowledge is represented has a powerful effect on the prospects for a computer or person to draw conclusions or make inferences from that knowledge. Consider the representation of numbers that we wish to add. Which is easier, adding 10 + 50 in Arabic numerals, or adding X plus L in Roman numerals? Consider also the use of algebraic symbols in solving problems for unknown numerical quantities, compared with trying to do the same problems just with words and numbers.

L

LISP
LISP (short for list processing language), a computer language, was invented by John McCarthy, one of the pioneers of artificial intelligence. The language is ideal for representing knowledge (e.g., If a fire alarm is ringing, then there is a fire) from which inferences are to be drawn.

M

Machine Learning
Machine learning refers to the ability of computers to automatically acquire new knowledge, learning from, for example, past cases or experience, from the computer's own experiences, or from exploration. Machine learning has many uses such as finding rules to direct marketing campaigns based on lessons learned from analysis of data from supermarket loyalty campaigns; or learning to recognize characters from people's handwriting. Machine learning enables computer software to adapt to changing circumstances, enabling it to make better decisions than non-AI software. Synonyms: learning, automatic learning.

Model-based Reasoning
Model-based reasoning (MBR) concentrates on reasoning about a system's behavior from an explicit model of the mechanisms underlying that behavior. Model-based techniques can very succinctly represent knowledge more completely and at a greater level of detail than techniques that encode experience, because they employ models that are compact axiomatic systems from which large amounts of information can be deduced.

Modeling
Synonym for simulation.

N

Natural Language Processing
English is an example of a natural language a computer language is not. For a computer to process a natural language, it would have to mimic what a human does. That is, the computer would have to recognize the sequence of words spoken by a person or another computer, understand the syntax or grammar of the words (i.e., do a syntactical analysis), and then extract the meaning of the words. A limited amount of meaning can be derived from a sequence of words taken out of context (i.e., by semantic analysis); but much more of the meaning depends on the context in which the words are spoken (e.g., who spoke them, under what circumstances, with what tone, and what else was said, particularly before the words), which would require a pragmatic analysis to extract. To date, natural language processing is poorly developed and computers are not yet able to even approach the ability of humans to extract meaning from natural languages; yet there are already valuable practical applications of the technology.

Neural Networks
Neural networks are an approach to machine learning which developed out of attempts to model the processing that occurs within the neurons of the brain. By using simple processing units (neurons), organized in a layered and highly parallel architecture, it is possible to perform arbitrarily complex calculations. Learning is achieved through repeated minor modifications to selected neurons, which results in a very powerful classification system. A problem with neural networks is that it very difficult to understand their internal reasoning process, and therefore to obtain an explanation for any particular conclusion. They are best used, therefore, when the results of a model are more important than understanding how the model works. Neural network software is used to recognize handwriting, and also to control chemical processes to run at desired conditions. Other applications include stock market analysis, fingerprint identification, character recognition, speech recognition, credit analysis, scientific analysis of data, and in neurophysiological research. Neural networks are also known as neural nets, connectionism, and parallel associative memory.

O

Object-oriented programming
An object-oriented problem-solving approach is very similar to the way a human solves problems. It consists of identifying objects and the correct sequence in which to use these objects to solve the problem. In other words, object-oriented problem solving consists of designing objects whose individual behaviors and interactions solve a specific problem. Interactions between objects take place through the exchange of messages, where a message to an object causes it to perform its operations and solve its part of the problem. The object-oriented

problem solving approach thus has four steps: 1) identify the problem; 2) identify the objects needed for the solution; 3) identify messages to be sent to the objects; and 4) create a sequence of messages to the objects that solve the problem.

In an object-oriented system, objects are data structures used to represent knowledge about physical things (e.g., pumps, computers, arteries, any equipment) or concepts (e.g., plans, designs, requirements). They are typically organized into hierarchical classes, and each class of object has information about its attributes stored in instance variables associated with each instance in the class. The only thing that an object knows about another object is that object's interface. Each object's data and logic is hidden from other objects. This allow the developer to separate an object's implementation from its behavior. This separation creates a "black-box" effect where the user is isolated from implementation changes. As long as the interface remains the same, any changes to the internal implementation is transparent to the user. Objects provide considerable leverage in representing the world in a natural way and in reusing code that operates on common classes of objects.

Ontology
A formal ontology is a rigorous specification of a set of specialized vocabulary terms and their relationships sufficient to describe and reason about the range of situations of interest in some domain.

In other words, it is a conceptual representation of the entities, events, and their relationships that compose a specific domain. Two primary relationships of interest are abstraction ("a cat is specific instance of a more general entity called animal") and composition ("a cat has whiskers and claws"). Ontologies are generally used to model a domain of interest, permitting inferential and deductive reasoning by learning systems.

P

Pattern Recognition
The use of feature analysis to identify an image of an object. May involve techniques such as statistical pattern recognition, Bayesian analysis, classification, cluster analysis, and analysis of texture and edges. See machine vision.

Plan Recognition
The goal of plan recognition is to interpret an agent's intentions by ascribing goals and plans to it based on partial observation of its behavior up to the current time. Divining the agent's underlying plan can be useful for many purposes including: interpreting the agent's past behavior, predicting the agent's future behavior, or acting to collaborate with (or thwart) the agent.

Planning and Scheduling
Planning is the field of AI that deals with the synthesis of plans, which are partial orders of (possibly conditional) actions to meet specified goals under specified constraints. It is related to scheduling, which is the task of determining when and with what resources to carry out each member of a specific set of actions to satisfy constraints regarding ordering, effectiveness and resource allocation. In 1991, SHAI developed the concept of intelligent entities for planning and scheduling applications. Intelligent entities play the role of managers of various resources, groups of resources, tasks, and projects made up of tasks.

Programming by Demonstration
Programming by demonstration (PBD) is a term that describes a variety of end-user programming techniques that generate code from examples provided by the user. The motivation behind Programming by Demonstration is simple and compelling - if a user knows how to perform a task on the computer, that alone should be sufficient to create a program to perform the task. It should not be necessary to learn a programming language like C++ or BASIC. The most simple version of Programming by Demonstration is accomplished by Macro Recorders which provide users with a way to record their actions. The user issues the "Record" command, performs a series of actions, and then issues the "Stop" command.

Prototyping
Prototyping is an important step in the development of a practical artificial intelligence application. An AI software prototype is usually a working piece of software that performs a limited set of the functions that the software designer envisages will be required by the user. It is used to convey to the users a clear picture of what is being developed to ensure that the software will serve the intended picture. An AI prototype, contrary to the practice with many other sorts of prototypes, is grown into the finished product, subject to changes at the request of the user. Unlike much other software, AI software cannot be subject to hard verification tests as it mirrors non-mathematical human reasoning, so the prototyping step provides necessary confirmation that the software will be "good enough" for its purpose at the expected cost.

Q

Qualitative Reasoning
Inexact reasoning, the opposite of quantitative reasoning . Also see **Commonsense Reasoning**.

R

Relevance Feedback
Relevance feedback methods are used in information retrieval systems to improve the results produced from a particular query by modifying the query based on the user's reaction to the initial retrieved documents. Specifically, the user's judgments of the relevance or non-relevance of some of the documents retrieved are used to add new terms to the query and to reweight query terms. For example, if all the documents that the user judges as relevant contain a particular term, then that term may be a good one to add to the original query.

Rule-based System
An **expert system** based on IF-THEN rules for representing knowledge.

S

Signal Filtering
Signal filtering is a techniques for removing the noise or static from a signal so the clear or underlying signal remains. This is a conventional technique commonly used by electrical engineers and others.

Simulated Annealing
Simulated annealing is an optimization method based on an analogy with the physical process of toughening alloys, such as steel, called annealing. Annealing involves heating an alloy and cooling it slowly to increase its toughness. In simulated annealing, an artificial "temperature" is used to control the optimization process, of finding the overall maximum or minimum of a function. As cooling a metal slowly allows the atoms time to move to the optimum positions for toughness, giving time to look for a solution in simulated annealing permits a successful search for the global optimum and avoids being trapped at a local suboptima. It is used, for example, to optimize routing of planes by airlines for most efficient use of the fleet. S. Kirkpatrick devised it.
Simulation
A simulation is a system that is constructed to work, in some ways, analogously to another system of interest. The constructed system is usually made simpler than the original system so that only the aspects of interest are mirrored. Simulations are commonly used to learn more about the behavior of the original system, when the original system is not available for manipulation. It may not be available because of cost or safety reasons, or it may not be built yet and the purpose of learning about it is to design it better. If the purpose of learning is to train novices, then cost, safety, or convenience are likely to be the reasons to work on a simulated system. The simulation may be a computer simulation (perhaps a realistic one of a nuclear power station's control room, or a mathematical one such as a spreadsheet for "what-if" analysis of a company's business); or it may be a small-scale physical model (such as a small-scale bridge, or a pilot chemical plant).

Statistical Learning
Statistical learning techniques attempt to construct statistical models of an entity based on surface features drawn from a large corpus of examples. These techniques generally operate independent of specific domain knowledge, training instead on a set of features that characterize an input example. In the domain of natural language, for example, statistics of language usage (e.g., word trigram frequencies) are compiled from large collections of input documents and are used to categorize or make predictions about new text. Systems trained through statistical learning have the advantage of not requiring human-engineered domain modeling. This strong dependence on the input corpus has the disadvantage of limiting their applicability to new domains, requiring access to large corpora of examples and a retraining step for each domain of interest. Statistical techniques thus tend to have high precision within a domain at the cost of generality across domains.

Supervised Learning
Organization and training of a neural network by a combination of repeated presentation of patterns, such as alphanumeric characters, and required knowledge. An example of required knowledge is the ability to recognize the difference between two similar characters such as O and Q. Synonym: learning with a teacher. Contrast with self-organized system; unsupervised learning.

T

Time Series Analysis
A time series is a sequence of observations of a particular variable over time (e.g., the daily closing level of Dow Jones Industrial Average). There is a wide range of statistical and temporal data mining techniques for analyzing such data. Two common uses for this type of analysis are forecasting future events (i.e., time series prediction) and searching a database of previous patterns for sequences that are similar to a particular pattern of interest. This is a conventional statistical technique.

Toy System
Small-scale implementation of a concept or model useful for testing a few main features, but unsuitable for complex or real-world problems. For example, a toy rule-based system may contain a few rules to construct an arch out of a number of pre-selected wooden blocks. It is a useful academic approach to unsolved problems. It is not employed in producing practical, real-world solutions.

Truth Maintenance Systems
Many conventional reasoning systems assume that reasoning is the process of deriving new knowledge from old, i.e., the number of things a person or intelligent software believes increases without retracting any existing knowledge, since known truths never change under this form of logic. This is called monotonic logic. However, this view does not accurately capture the way in which humans think since our actions constantly change what we believe to be true. Humans reason nonmonotonically, which means they reason based on incomplete or uncertain information, common sense, default values, changing conditions, and other assertions subject to retraction or revision. Truth maintenance systems seek to emulate the human reasoning process by recording reasons for our beliefs and reasons for retraction or revision of those beliefs, as well as the beliefs themselves. They are particularly useful in keeping track of goals, sub-goals, decisions, tasks, assignments, and design documents on complex projects (such as the design, construction, and testing of a major commercial aircraft) being undertaken by large numbers of people who may work for different organizations in different parts of the world. This is the sort of situation where a decision may be reversed, and not all the people who may have to react to that change may be properly informed. Project management software using TMS can help avoid design problems or wasted effort that can result from this sort of oversight. Also known as Reason Maintenance Systems.

References:

- Alan Bundy (ed.), *Artificial Intelligence Techniques: A Comprehensive Catalog*. 4th edition, Springer, 1997.
- **Artificial Intelligence Information** from PC AI magazine
- **Free On-Line Dictionary of Computing**